ELSEVIER

Theoretical Computer Science 275 (2002) 389–426

Theoretical Computer Science

www.elsevier.com/locate/tcs

Approximate reasoning by similarity-based SLD resolution

Maria I. Sessa

Dip. Matematica e Informatica, Universitá di Salerno, via S. Allende, I-84081 Baronissi (SA), Italy

Received February 2000; revised February 2001; accepted March 2001
Communicated by G. Levi

Abstract

In (Gerla and Sessa, Fuzzy Logic and Soft Computing, Kluwer, Norwell, 1999, pp. 19–31) a methodology that allows to manage uncertain and imprecise information in the frame of the declarative paradigm of Logic Programming has been proposed. With this aim, a Similarity relation $\mathscr{R}$ between function and predicate symbols in the language of a logic program is considered. Approximate inferences are then possible since similarity relation allows us to manage alternative instances of entities that can be considered "equal" with a given degree. The declarative semantics of the proposed transformation technique of logic programs is analyzed. The notion of fuzzy least Herbrand model is also introduced. In this paper the corresponding operational semantics is provided by introducing a modified version of SLD resolution. This top-down refutation procedure overcomes failure situations in the unification process by using the *similarity relation*. A generalized notion of most general unifier provides a numeric value which gives a measure of the exploited approximation. In this way, the SLD resolution is enhanced since it is possible both to handle uncertain or imprecise information, and to compute approximate answer substitutions, with an associated approximation-degree, when failures of the exact inference process occur. It can lead to the implementation of a more general PROLOG interpreter, without detracting from the elegance of the language.

Keywords: Logic programming; Approximate reasoning; SLD resolution; Similarity relation

1. Introduction

1.1. Motivations and previous works

The study of inference systems which allows us to deal with approximate reasoning is faced in literature with several and, often, very different approaches. Fuzzy Logic is a wide research area where theoretical issues and practical applications concerning these systems are crucial questions of interest. On the other hand, the pervading nature of the

E-mail address: misessa@unisa.it (M.I. Sessa).

0304-3975/02/$ - see front matter
PII: S0304-3975(01)00188-8

problem and its intrinsic difficulty caused the development of this kind of methodologies and tools also in different research fields as neural networks [28], genetic algorithms [13], evidence theory [34], probabilistic reasoning [30, 29, 23], weighted direct hypergraph [3], etc. In the past decades these research activities have been encouraged by an increasing number of real word applications (control systems, databases, expert systems, etc.); recently, theories which could manage uncertain and imprecise information are grouped in a new area which is nowadays referred to as Soft Computing.

In this paper we face the study of this problem in the framework of Logic Programming. In the early 1970s the idea that first order logic can be viewed as a programming language was introduced by Kowalski [22]. The main interest in this field traditionally concerns problems related to the analysis and the efficiency of exact inferences allowed by logic programs. However, very often the need of methods to enhance this capability, in order to deal with approximate information or flexible inference schemes, arises in many applications.

In general, approximate reasoning capabilities are introduced in the Logic Programming framework by considering the inference system based on fuzzy logic rather than on conventional two-valued logic. Several approaches extend the resolution rule to some fuzzy logic systems as possibilistic logic [10] and K-standard sequence logic [24]. In these proposals propositional or first order formulas are weighted with lower bound of possibility measures or with truth-value in $[0,1]$ and a model-theoretic semantics is provided. In [11] both model-theoretic and fixpoint semantics are presented for a particular case of inference, named quantitative reasoning, which also exploits the K-standard sequence logic. This approach is extended in [36] by allowing the representation of negative information. In [18] a general formal semantics for rule-based systems with uncertainty is presented. Function free rules of the form $p(x,\ldots):f(\alpha,\ldots,\gamma)\leftarrow a(y,\ldots):\alpha,\ \ldots,\ c(u,\ldots):\gamma$ are considered, where $p(x,\ldots)$, $a(y,\ldots),\ldots,c(u,\ldots)$ are usual literals, $\alpha,\ldots,\ \gamma$ are variable or constant values in $[0,1]$ representing certainty information about the associated literal, and f belongs to a finite set of functions. A generalization of the semi-naive bottom-up query evaluation procedure is also provided. In [19] a new semantics for such programs based on ideals of lattices is introduced, which extends the results in [24, 36]. In particular, it is proved that also formalisms for temporal reasoning fit into the proposed framework [1, 5, 35].

Several PROLOG interpreters based on fuzzy logic have been also presented in literature. The first implementation of the fuzzy resolution rule proposed in [24] is given in [17] with the PROLOG-Elf. In this system the inference takes care only of formulas with associated truth value greater than 0.5. Weighted rules are also exploited in the Fuzzy-PROLOG proposed in [27]. However, the heuristic technique which allows us to compute these weights, i.e. the truth values of the rules, hardly works and does not share the logical feature of the system. The language FPROLOG given in [26] tries to implement the capability of managing uncertain data represented by fuzzy sets. The basic idea is that some facts have associated truth value in $[0,1]$ representing their degree of membership in the set of true assertions. The truth value of a rule is computed by the truth values of its conditions, according to a given combi-

nation operator. A PROLOG interpreter based on Lukasiewicz logic, named LULOG and written in Common Lisp, is presented in [21]. Finally, for completeness sake, we also recall the fuzzy relational inference language FRIL presented in [4], which is a query language exploiting fuzzy relations, i.e., to any tuple t is associated a truth value $\chi(t) \in [0,1]$ denoting the degree to which t satisfies the relation. A fuzzy relational algebra is introduced. An operation in this algebra (as union, join, projection, etc.) takes one or more fuzzy relations as its operand(s) and produces a new fuzzy relation.

1.2. A new approach to approximate reasoning in Logic Programming

In [15] a new methodology that allows us to enhance the Logic Programming paradigm with approximate reasoning capability has been introduced. With respect to the previous literature, this approach is very different since the approximation is represented and managed at a syntactic-level, instead of at a rule-level. Roughly speaking, the basic idea is that the fuzziness feature is provided by an abstraction process which exploits a formal representation of similarity relations between elements in the alphabet of the language (constants, functions, predicates). On the contrary, in the underground logic theory, the inference rule as well as the usual crisp representation of the considered universe are not modified. It allows us to avoid both the introduction of weights on the clauses, and the use of fuzzy sets as elements of the language.

In this paper, the operational counterpart of this extension is faced by introducing a modified SLD resolution procedure. Such a procedure allows us to compute numeric values belonging to the interval $[0,1]$ which provides an approximation measure of the obtained solutions. These numeric values are computed through a generalized unification mechanism exploited in the top-down query evaluation process.

More precisely, the approach proposed in [15] starts by the observation that in a Logic Programming system if two constants or predicate names are different they represent distinct information, then no matching is possible. However, in many real world situations, an inference based on the equality between available values and required ones can produce a failure, but acceptable solutions could be reached by relaxing the exact-matching constraint. Indeed, human reasoning is often performed on the basis of "analogy" or "similarity" between entities and it induces an inference process based on an aware approximation.

Then, by following [37, 6], we consider reasonings that may be approximated by allowing the antecedent clauses of a rule to match its premises only approximately. An example is an inference concerning preferences on books such as the following informal one:

- *x is* ***thriller*** $\Rightarrow$ *x is* ***good*** *for me*
- *b is* ***adventurous***
- ***adventurous*** *is similar to* ***thriller*** *for me*

---

b is ***good*** *for me*.

Obviously, the similarity is a graded notion [38], so the degree at which we can admit the conclusion "*b is* ***good*** *for me*" depends on the degree of similarity between the predicates "***adventurous***" and "***thriller***". An important feature of such a kind of inference is the fact that the similarity is defined between symbols in the alphabet of the language, i.e. it is exploited at a syntactic-level. This makes our approach different from the usual one where the similarity is defined in the set of the interpretations (semantic-level) [31, 9, 12]. More resemblance can be found with a similarity-based approach proposed in the framework of relational databases [7], where the membership value of a tuple belonging to a response relation to a query is computed on the basis of a similarity defined over the domain.

The mathematical notion of *similarity relation* is a many valued extension of the equality, and it is widely exploited in any context where a weakening of the equality constraint is useful. In [15], the exact matching between different entities is relaxed by considering a similarity relation in the set of constant and predicate symbols in the language of a function free logic program P. Then, the program P is extended by adding new clauses which are "similar" at least with a fixed degree $\lambda \in [0,1]$ to the given ones. This program transformation is obtained by considering a suitable *closure operator* H_λ defined on sets of first order formulae. The new program $H_\lambda(P)$, named *extended-program*, allows us to enhance the inference process.

Several properties characterize the notion of similarity [20]. In particular, a well known result states that a similarity relation can be described "level by level" by a family of classical equivalence relations. Any equivalence relation of this family, named *cut of level* λ and denoted with $\cong_{\mathscr{R},\lambda}$, is obtained by considering as equivalent two elements which have a similarity value greater than a fixed $\lambda \in (0,1]$. Thus, in [15] an alternative transformation technique of logic programs has been also defined, by exploiting the abstraction provided by the relation $\cong_{\mathscr{R},\lambda}$, which synthesizes similarity properties of constants and predicate symbols of the program P. A preprocessing step transforms P in a new program P_λ, named *abstract-program*, and abstract computations are performed by taking into account elements in the quotient set of the equivalence relation $\cong_{\mathscr{R},\lambda}$. The equivalence between the two inference processes associated to the programs $H_\lambda(P)$ and P_λ has been proved by using an abstract interpretation technique and the notion of fuzzy least Herband model has been also introduced. The generalization of this approach to the case of program on a first order language with function symbols is provided in [32].

In this paper we face the procedural semantics of this extension of the Logic Programming paradigm. At first, in Section 5 we prove some properties concerning the computational equivalence of SLD resolution performed with respect to the extended and abstract programs $H_\lambda(P)$ and P_λ. In both cases the computation process is not modified at all.

Obviously, SLD resolution in the extended and abstract program needs some preprocessing steps in order to transform the given program P. On one hand, the effective constructions of the extended program $H_\lambda(P)$ can produce a very large program. On the other, the abstract program P_λ has a few number of clauses than P, but in any

case its effective construction requires memory and time resources. Thus, in Section 7 we define a new modified version of SLD resolution, named *similarity-based SLD*, which allows us to perform these kinds of extended computations exploiting the original program P, without any preprocessing steps. We prove some relations which link computed answer substitutions provided by similarity-based SLD resolution in P and solutions obtained by standard SLD resolution in the extended and abstract programs $H_\lambda(P)$ and P_λ. Relations with the declarative notions of fuzzy least Herbrand model are also discussed, which allow to state the computational equivalence between these procedures.

The proposed similarity-based SLD Refutation exploits a generalized notion of m.g.u., named *weak m.g.u.*. The failure of the unification between different function or predicate symbols is avoided by relaxing the equality constraint with the similarity relation. It leads to the notion of *unification-degree* associated to a substitution, and a weak m.g.u. is a more general substitution which provides the best unification-degree. These notions have been introduced in [14], where, by taking into account sets of symbols, a generalized unification algorithm based on similarity has been proposed. In this paper, a different unification algorithm is presented, which does not exploit sets of symbols and it is a simple modification of a standard algorithm. It allows us to overcome failures of the exact matching between function or predicate symbols if they are related by a non zero similarity value. By using this notion of weak m.g.u., when exact solutions do not exist, it is possible to obtain approximate computed answer substitutions with an associated *approximation-degree*.

In the sequel, we assume the leftmost selection rule whenever SLD resolution is considered. However, all the presented results can be analogously stated for any selection rule that does not depend on the function and predicate names and on the History of the derivation [2]. In particular, they hold for the safe version of the leftmost selection rule exploited in SLDNF resolution, and the classical fair rule which selects the atoms according to a queueing policy [25].

2. Preliminaries

Similarity relation is a mathematical notion that provides a way to manage alternative instances of an entity that can be considered "equal" with a given degree [38]. We summarize some results concerning this notion [20].

Let us recall that a *T-norm* $\wedge$ in $[0,1]$ is a binary operation $\wedge : [0,1] \times [0,1] \to [0,1]$ associative, commutative, nondecreasing in both the variables, and such that $x \wedge 1 = 1 \wedge x = x$ for any $x \in [0,1]$. In the sequel, we assume that $x \wedge y$ is the *minimum* between the two elements $x, y \in [0,1]$.

Definition 2.1. A *similarity* on a domain $\mathscr{U}$ is a fuzzy subset $\mathscr{R} : \mathscr{U} \times \mathscr{U} \to [0,1]$ of $\mathscr{U} \times \mathscr{U}$ such that the following properties hold:

(i) $\mathscr{R}(x,x) = 1$ for any $x \in \mathscr{U}$ (reflexivity)

(ii) $\mathscr{R}(x,y)=\mathscr{R}(y,x)$ for any $x,y\in\mathscr{U}$ (symmetry)
(iii) $\mathscr{R}(x,z)\geqslant\mathscr{R}(x,y)\wedge\mathscr{R}(y,z)$ for any $x,y,z\in\mathscr{U}$ (transitivity).
we say that $\mathscr{R}$ is *strict* if the following implication is also verified
(iv) $\mathscr{R}(x,z)=1\Rightarrow x=y$.

Similarity relations are strictly related with equivalence relations and, then, to closure operators, as stated by the following property.

Proposition 2.1. *Let $\mathscr{U}$ be a domain and $\mathscr{R}:\mathscr{U}\times\mathscr{U}\to[0,1]$ a similarity on $\mathscr{U}$. Then, for any $\lambda\in[0,1]$, the relation $\cong_{\mathscr{R},\lambda}$ in $\mathscr{U}$, named cut of level λ (in short λ-cut) of $\mathscr{R}$, defined as*

$$x\cong_{\mathscr{R},\lambda}y \quad\Leftrightarrow\quad \mathscr{R}(x,y)\geqslant\lambda,$$

is an equivalence relation. Also, the operator $H_\lambda:\mathscr{P}(\mathscr{U})\to\mathscr{P}(\mathscr{U})$ such that $\forall X\in\mathscr{P}(\mathscr{U})$

$$H_\lambda(X)=\{z\in\mathscr{U}\mid\exists x\in X:\ \mathscr{R}(z,x)\geqslant\lambda\}$$

is a closure operator.

Let us note that $H_\lambda(X)$ is given by the union of the equivalence classes modulo $\cong_{\mathscr{R},\lambda}$ of the elements $x\in X$. In the sequel, given $a\in\mathscr{U}$, the notation $H_\lambda(a)$ will be used instead of $H_\lambda(\{a\})$.

The notion of *λ-cut* allows us to define a similarity relation by means of a suitable family of equivalence relations according to the following result.

Proposition 2.2. *Let $\mathscr{R}$ be a similarity on a domain $\mathscr{U}$ and, for any $\lambda\in[0,1]$ let $\cong_{\mathscr{R},\lambda}$ be the λ-cut of $\mathscr{R}$. Then, $\{\cong_{\mathscr{R},\lambda}\}_{\lambda\in[0,1]}$ is a family of equivalence relations such that,*
(i) *for any μ and λ in $[0,1]$, $\lambda\leqslant\mu\Rightarrow\ \cong_{\mathscr{R},\lambda}\ \supseteq\ \cong_{\mathscr{R},\mu}$*
(ii) *for any μ in $[0,1]$, $\bigcap_{\lambda\leqslant\mu}\cong_{\mathscr{R},\lambda}=\cong_{\mathscr{R},\mu}$.*
Conversely, let $\{\cong_{\mathscr{R},\lambda}\}_{\lambda\in[0,1]}$ be a family of equivalence relations satisfying (i) *and* (ii)*. Then the relation $\mathscr{R}$ defined by setting*

$$\mathscr{R}(x,y)=Sup\{\lambda\in[0,1]\mid x\cong_{\mathscr{R},\lambda}y\}$$

is a similarity whose λ-cuts are the relations $\cong_{\mathscr{R},\lambda}$ belonging to the given family.

The equivalence $\cong_{\mathscr{R},\lambda}$ can be considered as a generalization of the identity relation. Proposition 2.1 is a crucial property in our approach. Indeed, the *λ-cut* relation $\cong_{\mathscr{R},\lambda}$ will be the formal tool exploited in the sequel in order to formalize the idea that two different constant symbols can be considered "equal" with a fixed tolerance level $\lambda\in(0,1]$. Such a level provides a measure of the allowed approximation in order to avoid failure of matching between different constant symbols.

Let us note that, when a similarity $\mathscr{R}$ is defined in a discrete domain $\mathscr{U}$, there exists only a discrete and ordered set of possible similarity values $\lambda_i\in[0,1]$, with i belonging to a set I of indexes. Hence, the family $\{\cong_{\mathscr{R},\lambda}\}_{\lambda\in[0,1]}$ in the previous proposition is

given by $\{\cong_{\mathscr{R},\lambda_i}\}_{i\in I}$. In the sequel we shall only deal with finite sets of indexes, but, since $\forall i \in I$ it is $\lambda_i \in [0,1]$, the obtained results can be extended to infinite sets of indexes.

The following example shows the idea that a similarity can be described by means of the family $\{\cong_{\mathscr{R},\lambda_i}\}_{1\leqslant i\leqslant n}$ of *λ-cut* relations.

Example 2.1. Let $U = \{a, b, c, d, e, a', b', c', d', e'\}$ be the set of elements arranged in the following table:

a, b	c
c'	a', b'

e	d
e'	d'

We can define a similarity $\mathscr{R}$ between elements in U by setting

$$\mathscr{R}(x, y) = 1 \quad \text{if } x = y,$$
$$\mathscr{R}(x, y) = 0.6 \quad \text{if } x, y \text{ are in the same small box},$$
$$\mathscr{R}(x, y) = 0.4 \quad \text{if } x, y \text{ are in the same big box},$$
$$\mathscr{R}(x, y) = 0 \quad \text{otherwise}.$$

As an example, by considering the cut relation $\cong_{\mathscr{R},0.3}$ of level $\lambda = 0.3$, the equivalence class of the element a is $\{a, b, c, a', b', c'\}$, whereas by considering the cut relation $\cong_{\mathscr{R},0.5}$ of level $\lambda = 0.5$, the equivalence class of the element a is $\{a, b\}$. According to the definition, it is $H_{0.3}(\{a, d\}) = U$ and $H_{0.5}(\{a, d\}) = \{a, b, d\}$.

An equivalent representation of $\mathscr{R}$ can be given by considering the quotient sets of the *λ-cuts* in the family $\{\cong_{\mathscr{R},\lambda_i}\}_{i\in I}$, corresponding to the different similarity levels $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\} = \{0, 0.4, 0.6, 1\}$.

$$U/_{\cong_{\mathscr{R},1}} = \{\{a\}, \{b\}, \{c\}, \{c'\}, \{a'\}, \{b'\}, \{e\}, \{d\}, \{e'\}, \{d'\}\},$$
$$U/_{\cong_{\mathscr{R},0.6}} = \{\{a, b\}, \{c\}, \{c'\}, \{a', b'\}, \{e\}, \{d\}, \{e'\}, \{d'\}\},$$
$$U/_{\cong_{\mathscr{R},0.4}} = \{\{a, b, c, c', a', b'\}, \{e, d, e', d'\}\},$$
$$U/_{\cong_{\mathscr{R},0}} = \{\{a, b, c, c', a', b', e, d, e', d'\}\} = \{U\}.$$

For any x, y in U, the similarity value $\mathscr{R}(x, y)$ can be obtained by considering the maximum level λ_i, $i \in \{1, 2, 3, 4\}$, such that the elements x and y belong to a same equivalence class in $\cong_{\mathscr{R},\lambda_i}$.

In the sequel we assume a familiarity with the basic notions of logic programming and abstract interpretation as stated in [2, 25, 8]. We briefly recall that a logic program P is a set of universally quantified Horn clauses on a first order language $\mathscr{L}$, denoted with $H \leftarrow B_1, \ldots, B_k$, and a goal is a negative clause, denoted with $\leftarrow A_1, \ldots, A_n$. The pre-order relation on substitutions $\preccurlyeq$ is such that $\sigma \preccurlyeq \tau$ if and only if there exists a substitution ρ with $\tau = \sigma\rho$. An expression E is an instance of an expression F if there

exists a substitution θ such that $E = F\theta$. When θ is a renaming of variables, we say that E is a variant of F. We denote with $B_{\mathscr{L}}$ the set of ground atomic formulae in $\mathscr{L}$, i.e. the Herbrand base of $\mathscr{L}$, and with T_P the immediate consequence operator $T_P : \mathscr{P}(B_{\mathscr{L}}) \to \mathscr{P}(B_{\mathscr{L}})$ defined by

$$T_P(X) = \{a \mid a \leftarrow a_1, \ldots, a_n \in \mathit{ground}(P) \text{ and } a_i \in X,\ n \geqslant i \geqslant 1\},$$

where $\mathit{ground}(P)$ denotes the set of all ground instances of clauses in P. The application of Tarski's fixpoint theorem yields a characterization of the semantics of P, which is the least Herbrand model M_P of P given by

$$M_P = \mathit{lfp}(T_P) = \bigcup_{n \geqslant 0} T_P^n(\emptyset).$$

Abstract interpretation is a theory of semantics approximation widely exploited to prove properties of programs written in any programming language. Roughly speaking, the correspondence between concrete and abstract properties is established by a pair of functions which is a Galois connection formalizing the loss of information. The notion of approximation is modeled by the *abstraction function* α that, for any concrete property $p^\flat \in P^\flat$ provides the best approximation $\alpha(p^\flat)$ in the abstract domain $P^\sharp$. The semantics of the abstract properties is given by the *concretization function* γ that for any abstract description $p^\sharp \in P^\sharp$ provides the corresponding concrete property $\gamma(p^\sharp)$ in the concrete domain $P^\flat$. The formal definition is as follows.

Definition 2.2. Let $P^\flat(\leqslant^\flat)$ and $P^\sharp(\leqslant^\sharp)$ be posets, a Galois connection is a pair of maps, $\alpha : P^\flat \to P^\sharp$ and $\gamma : P^\sharp \to P^\flat$, denoted with

$$P^\flat(\leqslant^\flat) \underset{\gamma}{\overset{\alpha}{\rightleftharpoons}} P^\sharp(\leqslant^\sharp)$$

such that for any $p^\sharp \in P^\sharp$ and $p^\flat \in P^\flat$,

$$\alpha(p^\flat) \leqslant^\sharp p^\sharp \quad \Leftrightarrow \quad p^\flat \leqslant^\flat \gamma(p^\sharp).$$

In the sequel we exploit a particular case of Galois connection, named *Galois surjection*, which is obtained when the abstraction function α is a surjection.

3. Logic programming with similarity

In [15] an extension of the declarative paradigm of the logic programming is proposed by considering similarity-based computations which allow us to perform approximate inferences. The notion of fuzzy least Herbrand model is also introduced. In [15] these notions are given for function free programs. In [32] the generalization to first order languages has been provided. In this section and in the following one, some properties concerning the main features of this approach are discussed.

Let $\mathscr{L}$ be a first order language and P a logic program in $\mathscr{L}$. In the classical case, function and predicate symbols of $\mathscr{L}$ are crisp elements, i.e., distinct elements represent distinct information and no matching is possible. In [15] we relax this constraint and suppose that it is possible to consider different functions, or different predicates with the same arity, as "similar" with a degree expressed by a value in $[0,1]$. In particular, equal elements have similarity degree 1 and completely different elements have similarity degree 0. This notion extends the usual equality relation and it is well modelled by the definition of similarity relation given in Section 2. More formally, let us denote with

V the set of *variable* symbols, ordered in a sequence $x_1, x_2, \ldots$, F the set of *function* symbols, R the set of *predicate* symbols.

As usual, constants can be considered as functions of arity zero. Let us consider a similarity relation $\mathscr{R}$ in $F \cup R \cup V$ such that $\mathscr{R}(t,t') = 0$ whenever one of the following cases occurs:

- t and t' are not both in R or F or V,
- t and t' are predicates in R with different arities,
- t and t' are functions in F with different arities,
- t and t' are variable and $t \neq t'$.

In other words, $\mathscr{R}$ provides a non-zero similarity value for function/predicate symbols with the same arity in $F \cup R$, whereas it is the identity relation for variables in V. In the sequel, the notation "/" is exploited when a proposition can be given for the elements both on the left and on the right of "/".

We can recursively extend this similarity to formulae in $\mathscr{L}$. Let $s_1, \ldots, s_n$, $r_1, \ldots, r_m$ be terms and f, g function/predicate symbols with arities n and m, respectively. If $n = m$, we set

$$\mathscr{R}(f(s_1, \ldots, s_n), g(r_1, \ldots, r_n)) = \mathscr{R}(f,g) \wedge \left(\bigwedge_{i=1}^{n} \mathscr{R}(s_i, r_i) \right),$$

otherwise, if $n \neq m$ the similarity value is zero. Then, let $C = A_0 \leftarrow A_1, \ldots, A_n$ and $C' = A'_0 \leftarrow A'_1, \ldots, A'_m$, be two Horn clauses. If $n = m$ we set

$$\mathscr{R}(C, C') = \bigwedge_{i=0}^{n} \mathscr{R}(A_i, A'_i)$$

otherwise, if $n \neq m$ we set $\mathscr{R}(C, C') = 0$.

The following proposition highlights useful properties of this extension of $\mathscr{R}$ that will be exploited in the sequel.

Proposition 3.1. *Given a first order language $\mathscr{L}$ and a similarity $\mathscr{R}$, then*

(i) *for any substitution θ and t, t' terms with $\mathscr{R}(t,t') \geqslant \lambda \in (0,1]$, it results that $\mathscr{R}(t\theta, t'\theta) = \mathscr{R}(t,t') \geqslant \lambda$.*

(ii) *if t, t' are terms (resp. atoms) with $\mathscr{R}(t,t') \geqslant \lambda \in (0,1]$ it follows that t, t', considered as strings of symbols, have the same length. Moreover, in corresponding*

positions they have either equal variable/bracket symbols, or pairs (s,s') *of function (resp. function/predicate) symbols such that* $\mathscr{R}(s,s') \geqslant \lambda$.

(iii) *if* C, C' *are clauses with* $\mathscr{R}(C,C') \geqslant \lambda \in (0,1]$ *it follows that* C, C', *considered as strings of symbols, have the same length. Moreover, in corresponding positions they have either equal variable/bracket/connective symbols, or pairs* (s,s') *of function/predicate symbols such that* $\mathscr{R}(s,s') \geqslant \lambda$.

Proof. (ii), (iii) By definition of similarity between terms, atoms, clauses.

(i) Two cases are possible. If t and t' are both ground, then θ does not affect t and t' then the thesis is true. Otherwise, by (ii) it follows that t and t' have equal variables in corresponding positions. In this case, equal variables are substituted with equal terms by θ. Since equal terms have similarity value 1, the thesis follows by the definition of similarity between terms (resp. atoms). □

According to Proposition 2.1, let $\cong_{\mathscr{R},\lambda}$ be the equivalence relation, named cut of level λ of $\mathscr{R}$ in $F \cup R \cup V$, $\lambda \in (0,1]$, defined as

$$x \cong_{\mathscr{R},\lambda} y \quad \Leftrightarrow \quad \mathscr{R}(x,y) \geqslant \lambda, \quad \text{with } x, y \in F \cup R \cup V.$$

Following the same recursive steps exploited to extend the similarity $\mathscr{R}$, at first the equivalence relation $\cong_{\mathscr{R},\lambda}$ can be extended to terms/atoms in $\mathscr{L}$ with the same arity by setting

$$f(s_1,\ldots,s_n) \cong_{\mathscr{R},\lambda} g(r_1,\ldots,r_n) \quad \Leftrightarrow \quad f \cong_{\mathscr{R},\lambda} g \quad \text{and} \quad s_i \cong_{\mathscr{R},\lambda} r_i, \quad n \geqslant i \geqslant 1$$

then, to clauses in $\mathscr{L}$ with the same number of literals by setting

$$C \cong_{\mathscr{R},\lambda} C' \quad \Leftrightarrow \quad A_i \cong_{\mathscr{R},\lambda} A'_i, \quad n \geqslant i \geqslant 0.$$

In parallel, the closure operator H_λ associated to $\cong_{\mathscr{R},\lambda}$ can be defined on sets of terms, atoms and clauses in $\mathscr{L}$, too. In particular, by denoting with □ the empty clause, it is $H_\lambda(\square) = \square$.

Then, if P is a logic program and $\lambda \in (0,1]$, the set

$$H_\lambda(P) = \{C' \in \mathscr{L} \mid \exists C \in P\colon \mathscr{R}(C,C') \geqslant \lambda\}$$

is constructed by adding to P all the clauses in $\mathscr{L}$ which are equivalent to clauses in P modulo $\cong_{\mathscr{R},\lambda}$, i.e., which can be obtained by replacing function and predicate symbols of a clause in P with symbols having similarity degree greater or equal to λ. Also $H_\lambda(P)$ is a logic program, that we name *extended-program of level* λ.

Thus, a first way to manage the weakening of the equality relation between function/predicate symbols, expressed by the similarity relation $\mathscr{R}$, can be given by considering inferences with respect to $H_\lambda(P)$. Denoted with $T_{H_\lambda(P)}$ the immediate consequence operator of the program $H_\lambda(P)$, the least Herbrand model of $H_\lambda(P)$ is

$$M_{H_\lambda(P)} = lfp(T_{H_\lambda(P)}) = \bigcup_{n \geqslant 0} T^n_{H_\lambda(P)}(\emptyset).$$

An alternative way to manage the information carried on by the similarity introduced between function/predicate symbols in P, can be given by identifying symbols which have similarity degree greater or equal to λ. In other word, we consider the quotient sets $F/_{\cong_{\mathscr{R},\lambda}}$ and $R/_{\cong_{\mathscr{R},\lambda}}$ as new sets of function and predicate symbols, respectively. We denote with $\mathscr{L}_\lambda$ the first order language related to the new resulting alphabet. A link between the two languages $\mathscr{L}$ and $\mathscr{L}_\lambda$ is stated by defining the function $\tau_\lambda : \mathscr{L} \to \mathscr{L}_\lambda$, named *translation up to* $\cong_{\mathscr{R},\lambda}$, that associates to a formula G in $\mathscr{L}$, the formula $\tau_\lambda(G)$ in $\mathscr{L}_\lambda$ obtained by replacing predicate and function symbols in G with their equivalence classes in $F/_{\cong_{\mathscr{R},\lambda}}$ and $R/_{\cong_{\mathscr{R},\lambda}}$, respectively.

More formally we give the following definition.

Definition 3.1. Given a first order language $\mathscr{L}$, a similarity $\mathscr{R}$ and $\lambda \in (0,1]$, we define the function $\tau_\lambda : F \cup R \cup V \to F/_{\cong_{\mathscr{R},\lambda}} \cup R/_{\cong_{\mathscr{R},\lambda}} \cup V$ as follows:

$$\tau_\lambda(x) = x \quad \text{for any variable } x \in V,$$

$$\tau_\lambda(f) = [f] \quad \text{for any function/predicate symbol } f \in F \cup R.$$

Recursively, we define the extension $\tau_\lambda : \mathscr{L} \to \mathscr{L}_\lambda$ to the sets of formulae in $\mathscr{L}$ by setting $\tau_\lambda(\Box) = \Box$ and:

- $\forall t_1, \ldots, t_n$ terms in $\mathscr{L}$, and f function/predicate symbol in $F \cup R$

$$\tau_\lambda(f(t_1, \ldots, t_n)) = \tau_\lambda(f)(\tau_\lambda(t_1), \ldots, \tau_\lambda(t_n)),$$

- $\forall A, B$ fomulae in $\mathscr{L}$:

$$\tau_\lambda(A \wedge B) = \tau_\lambda(A) \wedge \tau_\lambda(B), \quad \tau_\lambda(A \vee B) = \tau_\lambda(A) \vee \tau_\lambda(B),$$

$$\tau_\lambda(\neg A) = \neg\tau_\lambda(A), \quad \tau_\lambda(\forall A) = \forall\tau_\lambda(A), \quad \tau_\lambda(\exists A) = \exists\tau_\lambda(A).$$

The following proposition highlights some properties of τ_λ that will be exploited in the sequel.

Proposition 3.2. *For any $\lambda \in (0,1]$ we have that*

(i) *given a term/atom t, then $t' \in H_\lambda(t)$ if and only if $\tau_\lambda(t) = \tau_\lambda(t')$,*

(ii) *given a clause G, then $G' \in H_\lambda(G)$ if and only if $\tau_\lambda(G) = \tau_\lambda(G')$,*

(iii) *if t and T are terms (resp. atoms) such that $T = \tau_\lambda(t)$, then T and t considered as strings have the same length. Moreover, in corresponding positions they have either equal variable/bracket symbols, or a pair (S,s) of function (resp. function/predicate) symbols such that $S = \tau_\lambda(s)$,*

(iv) *if C and C' are clauses such that $C' = \tau_\lambda(C)$, then C' and C considered as strings have the same length. Moreover, in corresponding positions they have either equal variable/bracket/connective symbols, or a pair (S,s) of function/predicate symbols such that $S = \tau_\lambda(s)$.*

Proof. (i) Any term/atom $t' \in H_\lambda(t)$ is such that $\mathscr{R}(t',t) \geqslant \lambda \in (0,1]$. Then, the thesis follows by Proposition 3.1(ii) and by Definition 3.1.

(ii) Any clause $G' \in H_\lambda(t)$ is such that $\mathscr{R}(G', G) \geqslant \lambda \in (0,1]$. Then, the thesis follows by Proposition 3.1(iii) and by Definition 3.1.

(iii), (iv) By Definition 3.1. □

Then, if P is a logic program on the language $\mathscr{L}$ and $\lambda \in (0,1]$, the set

$$P_\lambda = \tau_\lambda(P) = \{K \in \mathscr{L}_\lambda \mid \exists H \in P\colon \tau_\lambda(H) = K\}$$

is obtained by replacing function and predicate symbols of a clause in P with the related equivance classes modulo $\cong_{\mathscr{R},\lambda}$. As a consequence, also P_λ is a logic program, that we name *abstract-program of level* λ. It is worth to explicitly note that $H_\lambda(P)$ is a program on the same language of P and, in general, it is bigger than P; on the contrary, P_λ is a program on a different language and, in general, it is smaller.

Thus, the program P_λ could be used to manage similarity-based reasoning as well as $H_\lambda(P)$. Let $B_{\mathscr{L}_\lambda}$ denote the Herbrand base of $\mathscr{L}_\lambda$, and T_{P_λ} the immediate consequence operator of P_λ, the least Herbrand model of P_λ is given by

$$M_{P_\lambda} = lfp(T_{P_\lambda}) = \bigcup_{n \geqslant 0} T^n_{P_\lambda}(\emptyset).$$

By exploiting an abstract interpretation technique, in [15] the equivalence of these two approaches has been shown for processing information provided by a similarity relation defined in the language of a function free program P. At first, it is shown that a Galois connection can be defined by considering as concrete and abstract domains the power sets of the languages $B_{\mathscr{L}}$ and $B_{\mathscr{L}_\lambda}$, respectively, as stated by the following proposition.

Proposition 3.3. *Given a function free language* $\mathscr{L}$, *a similarity* $\mathscr{R}$ *and* $\lambda \in (0,1]$. *The pair of functions* $\alpha : \mathscr{P}(B_{\mathscr{L}}) \to \mathscr{P}(B_{\mathscr{L}_\lambda})$ *and* $\gamma : \mathscr{P}(B_{\mathscr{L}_\lambda}) \to \mathscr{P}(B_{\mathscr{L}})$ *such that*, $\forall X \in \mathscr{P}(\mathscr{L})$, *and* $\forall Y \in \mathscr{P}(\mathscr{L}_\lambda)$

$$\alpha(X) = \tau_\lambda(X), \quad \gamma(Y) = \tau_\lambda^{-1}(Y)$$

provides a Galois surjection

$$(\mathscr{P}(B_{\mathscr{L}}), \subseteq) \underset{\gamma}{\overset{\alpha}{\rightleftharpoons}} (\mathscr{P}(B_{\mathscr{L}_\lambda}), \subseteq),$$

between the complete lattices $(\mathscr{P}(B_{\mathscr{L}}), \subseteq)$ *and* $(\mathscr{P}(B_{\mathscr{L}_\lambda}), \subseteq)$ *with* $\gamma \circ \alpha = H_\lambda$.

Then, the α-optimality of the associated operators $T_{H_\lambda(P)}$ and T_{P_λ} is proven. It allows us to state the α-optimality of the abstract semantics, i.e., to relate the least fixpoints by means of the abstraction function α. In [32] this result has been extended to programs with function symbols by considering a more general definition of translation function τ_λ and the extended program $H_\lambda(ground(P))$. Then, since it is easy to prove that $M_{H_\lambda(ground(P))} = M_{H_\lambda(P)}$, the following result can be stated.

Theorem 3.1. *Given a program P on a first order language $\mathscr{L}$, the abstract semantics of the Galois surjection given in Proposition* 3.3 *is α-optimal with respect to the immediate consequence operators, i.e.*

$$\alpha(M_{H_\lambda(P)}) = \alpha(lfp(T_{H_\lambda(P)}) = lfp(T_{P_\lambda}) = M_{P_\lambda}.$$

The previous theorem allows us to highlight the main feature of our approach. Indeed, the basic idea is that the approximation of the exact reasoning is performed by representing and managing fuzziness through an abstraction process grounded on similarity relations between elements of the language (constants, functions, predicates). The α-optimality property ensures that abstract computations provide the corresponding concrete semantics without loss of information.

Finally, the following proposition shows that higher values of the fixed similarity level λ corresponds to lower possibility of inferences.

Proposition 3.4. *Given a similarity $\mathscr{R}$, a program P on a first order language $\mathscr{L}$ and $\lambda, \eta \in (0,1]$ with $\eta \prec \lambda$, then $M_{H_\eta(P)} \supseteq M_{H_\lambda(P)}$.*

Proof. If C is a clause belonging to $H_\lambda(P)$, by Proposition 2.1 there is a clause $C' \in P$ such that $\mathscr{R}(C, C') \geqslant \lambda \succ \eta$. Then C also belongs to $H_\eta(P)$, i.e. $H_\eta(P) \supseteq H_\lambda(P)$. The thesis follows since logic programming is a monotone inference system. □

4. Fuzzy least Herbrand model

Proposition 3.4 highlights that a graded notion of logical consequence can be considered by taking into account the different levels $\lambda \in (0,1]$ which provide the extended programs $H_\lambda(P)$. Then, in a natural way, the crisp notion of least Herbrand model can be fuzzified as well.

Let us recall that a fuzzy subset of a domain U is defined by giving its membership function $f : U \to [0,1]$. Such a function is a generalization of the characteristic function of a crisp subset, i.e., a greater value of $f(x)$, $x \in U$ corresponds to an higher membership level of the element x to the considered fuzzy subset. Thus, in [15] the notion of *fuzzy least Herbrand model* of a logic program P has been introduced as follows.

Definition 4.1. Let P be a logic program on a first order language $\mathscr{L}$ with a similarity $\mathscr{R}$, and $\{M_{H_\lambda(P)}\}_{\lambda \in [0,1]}$ the family of Herbrand models defined in Section 3. The *fuzzy least Herbrand model* $M_{P,\mathscr{R}} : B_{\mathscr{L}} \to [0,1]$ of the program P with respect to the similarity $\mathscr{R}$ is defined by setting, for any $L \in B_{\mathscr{L}}$,

$$M_{P,\mathscr{R}}(L) = \mathrm{Sup}\{\lambda \in [0,1] \mid L \in M_{H_\lambda(P)}\}$$

or, equivalently,

$$M_{P,\mathscr{R}}(L) = \mathrm{Sup}\{\lambda \in [0,1] \mid H_\lambda(P) \models L\}.$$

Exploiting the α-optimality of the Galois connection defined in the previous section, the following alternative characterization of the fuzzy least Herbrand model $M_{P,\mathscr{R}}$ can be given.

Theorem 4.1. *Let P be a logic program on a first order language $\mathscr{L}$ with a similarity $\mathscr{R}$. Then, for any $L \in B_{\mathscr{L}}$,*

$$M_{P,\mathscr{R}}(L) = \mathrm{Sup}\{\lambda \in [0,1] \mid \tau_\lambda(L) \in M_{P_\lambda}\}.$$

or, equivalently,

$$M_{P,\mathscr{R}}(L) = \mathrm{Sup}\{\lambda \in [0,1] \mid P_\lambda \models \tau_\lambda(L)\}.$$

Thus, according to the previous results, we conclude that, in order to compute the fuzzy least Herbrand model of a program P with a similarity $\mathscr{R}$, we can equivalently perform our computations in the extended or in the abstract domain.

We recall that the family $\{\cong_{\mathscr{R},\lambda_i}\}_{1 \leqslant i \leqslant n}$ provides all the different *λ-cuts* associated to the similarity $\mathscr{R}$ and, then, all the different closure operator H_λ. The following proposition provides the bases for defining an algorithm for a bottom-up computation of the fuzzy least Herbrand model.

Proposition 4.1. *Let P be a logic program on a first order language $\mathscr{L}$ with a similarity $\mathscr{R}$, and $\lambda_1 \prec \lambda_2 \prec \cdots \prec \lambda_n$, with $\lambda_i \in [0,1]$, $1 \leqslant i \leqslant n$, all the different similarity values in $\mathscr{R}$. Then, for any $L \in B_{\mathscr{L}}$*

$$M_{P,\mathscr{R}}(L) = \max_{1 \leqslant i \leqslant n} \{\lambda_i \mid L \in M_{H_{\lambda_i}(P)}\}.$$

Proof. Since $\lambda_1 \prec \lambda_2 \prec \cdots \prec \lambda_n$, by Proposition 3.4 we have that

$$M_{H_{\lambda_1}(P)} \supseteq M_{H_{\lambda_2}(P)} \supseteq \cdots \supseteq M_{H_{\lambda_n}(P)}.$$

The least Herbrand models appearing in the previous relation are all the possible different ones that can be obtained by considering the extended programs $H_\lambda(P)$, with $\lambda \in [0,1]$. Then the thesis follows by Definition 4.1 of fuzzy least Herbrand model. □

An analogous result can be stated by considering the least Herbrand model of the abstract program P_λ.

Proposition 4.2. *Let P be a logic program on a first order language $\mathscr{L}$ with a similarity $\mathscr{R}$, and $\lambda_1 \prec \lambda_2 \prec \cdots \prec \lambda_n$, with $\lambda_i \in [0,1]$, $1 \leqslant i \leqslant n$, all the different similarity values in $\mathscr{R}$. Then, for any $L \in B_{\mathscr{L}}$*

$$M_{P,\mathscr{R}}(L) = \max_{1 \leqslant i \leqslant n} \{\lambda_i \mid \tau_{\lambda_i}(L) \in M_{P_{\lambda_i}}\}.$$

Proof. By Proposition 4.1, for any $L \in B_{\mathscr{L}}$ it is $M_{P,\mathscr{R}}(L) = \lambda_j$, with $\lambda_j = \max_{1 \leqslant i \leqslant n}\{\lambda_i \mid L \in M_{H_{\lambda_i}(P)}\}$. Then, $\tau_{\lambda_j}(L) \in \tau_{\lambda_j}(M_{H_{\lambda j}(P)})$. By Proposition 3.3 there exists a Galois surjection

$$(\mathscr{P}(B_{\mathscr{L}}), \subseteq) \underset{\gamma}{\overset{\alpha}{\rightleftharpoons}} (\mathscr{P}(B_{\mathscr{L}_\lambda}), \subseteq)$$

with $\alpha = \tau_{\lambda_j}$. Then, since Theorem 3.1 implies that $\tau_{\lambda_j}(M_{H_{\lambda j}(P)}) = M_{P_{\lambda j}}$, the thesis is proved. □

Finally, let us state the following relation between the standard least Herbrand Model of a program P and the introduced fuzzy generalization of this notion.

Proposition 4.3. *Let P be a logic program on a first order language $\mathscr{L}$ with a strict similarity $\mathscr{R}$. Then, denoted with M_P the least Herbrand model of P, it is $L \in M_P$ if and only if $M_{P,\mathscr{R}}(L) = 1$.*

Proof. Since $\mathscr{R}$ is a strict similarity, for $\lambda = 1$ it is $H_1(P) = P$. Then, if $L \in M_P = M_{H_1(P)}$, by Proposition 4.1 it follows that $M_{P,\mathscr{R}}(L) = 1$. Conversely, if $M_{P,\mathscr{R}}(L) = 1$, by Definition 4.1 it is $L \in M_{H_1(P)} = M_P$. □

5. SLD resolution in the extended and abstract programs

The transformation of a program P into the extended or abstract program generally changes the semantics of P. Indeed, when the extended-program $H_\lambda(P)$ is considered, new clauses on the language $\mathscr{L}$ are added to P. This is a straight way to manage information provided by the introduced similarity, and the usual SLD refutation in $H_\lambda(P)$ directly provides the related procedural semantics.

On the other hand, the abstract program P_λ allows us to represent, in a synthetic way, information provided by the similarity by means of the equivalence classes of the *λ-cut* relation. Also in this case, a procedural semantics can be given by considering a simple modification of an SLD interpreter. More precisely, for a fixed value $\lambda \in (0, 1]$, at first a preprocessing step provides the elements in the quotient sets $F/\approxeq_{\mathscr{R},\lambda}$ and $R/\approxeq_{\mathscr{R},\lambda}$. By exploiting the translation function τ_λ, the set of clauses $P \cup \{G_0\}$ can be transformed in the set of clauses $\tau_\lambda(P \cup \{G_0\}) = P_\lambda \cup \{\tau_\lambda(G_0)\}$. Then, the SLD interpreter computes with respect to the abstract program P_λ and goal $\tau_\lambda(G_0)$. It is worth stressing that, also in this case, the interpreter performs SLD refutations without changes in the usual derivation process.

The following results state some relations between these SLD resolution processes associated to the extended and abstract programs. We assume the leftmost selection rule whenever SLD resolution is considered. However, all the presented results can be analogously stated for any selection rule that does not depend by the function and predicate names and by the History of the derivation [2]. In particular, they hold for

the safe version of the leftmost selection rule exploited in SLDNF resolution, and the classical fair rule which selects the atoms following a queueing policy [25].

Proposition 5.1. *Let $\mathscr{R}$ be a similarity, P a program on a first order language, G_0 a goal. If*

$$D = G_0 \Rightarrow_{C_1,\theta_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m} G_m$$

is a SLD derivation for $H_\lambda(P) \cup \{G_0\}$ with $\sigma = \theta_1 \ldots \theta_m = \{x_1/u_1, \ldots, x_k/u_k\}$, then there exists an SLD derivation for $P_\lambda \cup \{\tau_\lambda(G_0)\}$

$$D' = \tau_\lambda(G_0) \Rightarrow_{C'_1,\theta'_1} G'_1 \Rightarrow \cdots \Rightarrow_{C'_m,\theta'_m} G'_m,$$

where $\sigma' = \theta'_1 \ldots \theta'_m = \{x_1/T_1, \ldots, x_k/T_k\}$, with $T_h = \tau_\lambda(u_h)$, $k \geqslant h \geqslant 1$ and $G'_i = \tau_\lambda(G_i)$, $C'_i = \tau_\lambda(C_i)$, $m \geqslant i \geqslant 0$.

Proof. We prove the thesis by induction on the length of D. If the length of D is zero the thesis is true. Let us suppose that the thesis is true for length of D equal m. Let

$$D = G_0 \Rightarrow_{C_1,\theta_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m} G_m \Rightarrow_{C_{m+1},\theta_{m+1}} G_{m+1}$$

be an existing SLD derivation for $H_\lambda(P) \cup \{G_0\}$ of length $m+1$ where $\theta_1 \ldots \theta_m = \{x_1/v_1, \ldots, x_r/v_r\}$. By the inductive hypothesis there exists an SLD derivation

$$D'_1 = \tau_\lambda(G_0) \Rightarrow_{C'_1,\theta'_1} G'_1 \Rightarrow \cdots G'_m \Rightarrow_{C'_m,\theta'_m}$$

for $P_\lambda \cup \{\tau_\lambda(G_0)\}$, where $\theta'_1 \ldots \theta'_m = \{x_1/\tau_\lambda(v_1), \ldots, x_r/\tau_\lambda(v_r)\}$ and $G'_i = \tau_\lambda(G_i)$, $m \geqslant i \geqslant 1$.

Let us denote with A the head of the standardized input clause $C_{m+1} = A \leftarrow B_1 \ldots B_j$ and with L the selected atom of G_m in D. Since by inductive hypothesis $G'_m = \tau_\lambda(G_m)$, by Definition 3.1 the leftmost atom L' in G'_m is such that $L' = \tau_\lambda(L)$. Then, by Proposition 3.2(iii) L' and L are strings of the same length that in corresponding positions have or equal variables/brackets or a pair (S, s) of function/predicate symbols, with $S = \tau_\lambda(s)$. If we consider $A' = \tau_\lambda(A)$, the same kind of correspondences hold also for the atoms A' and A. Then, the unification algorithm on L' and A' emulates the behavior on L and A, provided that any function/predicate symbol s in L and A is substituted by its equivalence class $S = \tau_\lambda(s)$ modulo $\approxeq_{\mathscr{R},\lambda}$. As a consequence, if $\theta_{m+1} = \{x_1/w_1, \ldots, x_l/w_l\}$, it follows that $\theta'_{m+1} = \{x_1/\tau_\lambda(w_1), \ldots, x_l/\tau_\lambda(w_l)\}$ is an m.g.u. for L' and A'. This construction and the inductive hypothesis imply that, if $\sigma = \theta_1 \ldots \theta_{m+1} = \{x_1/u_1, \ldots, x_k/u_k\}$, it results $\sigma' = \theta'_1 \ldots \theta'_{m+1} = \{x_1/\tau_\lambda(u_1), \ldots, x_k/\tau_\lambda(u_k)\}$.

We observe that, w.r.t. D'_1, the clause $C'_{m+1} = \tau_\lambda(C_{m+1}) = \tau_\lambda(A) \leftarrow \tau_\lambda(B_1) \ldots \tau_\lambda(B_j)$ is a standardization apart of a clause in P_λ. Since by the inductive hypothesis it is $G'_m = \tau_\lambda(G_m)$, by exploiting in D'_1 as input clause C'_{m+1} and m.g.u. θ'_{m+1}, we obtain a resolvent $G'_{m+1} = \tau_\lambda(G_{m+1})$. Thus the thesis is proved. □

We note that it is always possible to assume that the alphabet of $\mathscr{L}$ is well ordered, and to consider the induced well ordering in $\mathscr{L}$. Then, given an element S in the

alphabet of $\mathscr{L}_\lambda$, i.e., an equivalence class of an element in $\mathscr{L}$, we can denote with $\phi(S)$ the first element in S. In such a way we obtain an injective function $\phi : F/\cong_{\mathscr{R},\lambda} \cup R/\cong_{\mathscr{R},\lambda} \to F \cup R$, which for any equivalence class provides a fixed representative element. Obviously, $\tau_\lambda(\phi(S)) = S$, for any S.
We can recursively extend ϕ to terms, atoms and formulae in $\mathscr{L}_\lambda$ by setting $\phi(\Box) = \Box$ and

- for any function/predicate symbol S in $F/\cong_{\mathscr{R},\lambda} \cup R/\cong_{\mathscr{R},\lambda}$ and $T_1, \ldots, T_n$ terms in $\mathscr{L}_\lambda$

$$\phi(S(T_1, \ldots, T_n)) = \phi(S)(\phi(T_1), \ldots, \phi(T_n)),$$

- $\forall A, B$ formulae in $\mathscr{L}_\lambda$:

$$\phi(A \wedge B) = \phi(A) \wedge \phi(B), \quad \phi(A \vee B) = \phi(A) \vee \phi(B),$$

$$\phi(\neg A) = \neg\phi(A), \quad \phi(\forall A) = \forall\phi(A), \quad \phi(\exists A) = \exists\phi(A).$$

It is worth stressing that if T is a term in $\mathscr{L}_\lambda$, then $\phi(T) = u$ is the minimum (with respect to the induced well ordering in $\mathscr{L}$) in the equivalence class of the terms u' in $\mathscr{L}$ such that $\tau_\lambda(u') = T$. In the following proposition, this particular function ϕ is exploited.

Proposition 5.2. *Let $\mathscr{R}$ be a similarity, P a logic program on a first order language $\mathscr{L}$ and G_0 a goal. If there exists a SLD derivation for $P_\lambda \cup \{\tau_\lambda(G_0)\}$*

$$D' = \tau_\lambda(G_0) \Rightarrow_{C'_1,\theta'_1} G'_1 \Rightarrow \cdots \Rightarrow_{C'_m,\theta'_m} G'_m$$

where $\sigma' = \theta'_1 \ldots \theta'_m = \{x_1/T_1, \ldots, x_k/T_k\}$, then there exists a SLD-derivation for $H_\lambda(P) \cup \{G_0\}$

$$D = G_0 \Rightarrow_{C_1,\theta_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m} G_m$$

where $\sigma = \theta_1 \ldots \theta_m = \{x_1/u_1, \ldots, x_k/u_k\}$, with $T_h = \tau_\lambda(u_h)$, $k \geqslant h \geqslant 1$, $G'_i = \tau_\lambda(G_i)$, $C'_i = \tau_\lambda(C_i)$, $m \geqslant i \geqslant 0$. Moreover, it is $\phi(T_h) = u_h$, $k \geqslant h \geqslant 1$, and $\phi(G'_i) = G_i$, $\phi(C'_i) = C_i$, $m \geqslant i \geqslant 0$.

Proof. The proof line is the same as in Proposition 5.1. By induction on the length of D'. If the length of D' is zero the thesis is true. Let us suppose that the thesis is true for length of D' equal m. Let

$$D' = \tau_\lambda(G_0) \Rightarrow_{C'_1,\theta'_1} G'_1 \Rightarrow \cdots \Rightarrow_{C'_m,\theta'_m} G'_m \Rightarrow_{C'_{m+1},\theta'_{m+1}} G'_{m+1}$$

be an existing SLD derivation for $P_\lambda \cup \{\tau_\lambda(G_0)\}$ of length $m+1$ where $\theta'_1 \ldots \theta'_m = \{x_1/V_1, \ldots, x_r/V_r\}$. By the inductive hypothesis there exists an SLD derivation

$$D_1 = G_0 \Rightarrow_{C_1,\theta_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m} G_m$$

for $P_\lambda \cup \{G_0\}$, where $\theta_1 \ldots \theta_m = \{x_1/v_1, \ldots, x_r/v_r\}$, with $V_h = \tau_\lambda(v_h)$, $r \geqslant h \geqslant 1$, and $G'_i = \tau_\lambda(G_i)$, $C'_i = \tau_\lambda(C_i)$, $m \geqslant i \geqslant 0$. Moreover, it is $\phi(V_h) = v_h$, $r \geqslant h \geqslant 1$, and $\phi(G'_i) = G_i$, $\phi(C'_i) = C_i$, $m \geqslant i \geqslant 0$.

Let us denote with A' the head of the standardized input clause $C'_{m+1} = A' \leftarrow B'_1 \dots B'_j$ and with L' the selected atom of G'_m in D'. Since by inductive hypothesis $G'_m = \tau_\lambda(G_m)$ with $\phi(G'_i) = G_i$, by Definition 3.1 the leftmost atom L in G_m is such that $L' = \tau_\lambda(L)$ and $\phi(L') = L$. Then, by Proposition 3.2(iii) L' and L are strings of the same length and in corresponding positions they have or equal variables/brackets or a pair (S, s) of function/predicate symbols with $S = \tau_\lambda(s)$ and $\phi(S) = s$. If we consider $A = \phi(A')$, it is $A' = \tau_\lambda(A)$, then the same kind of correspondence holds also for the atoms A' and A. Then, the unification algorithm on L and A emulates the behavior on L' and A', provided that any function/predicate symbol S in L and A is substituted by its fixed representative element $\phi(S)$. As a consequence, if $\theta'_{m+1} = \{x_1/W_1, \dots, x_l/W_l\}$, then $\theta_{m+1} = \{x_1/\phi(W_1), \dots x_l/\phi(W_l)\}$ is an m.g.u. for L and A. This construction and the inductive hypothesis imply that, if $\sigma' = \theta'_1 \dots \theta'_{m+1} = \{x_1/T_1, \dots, x_k/T_k\}$, it results in $\sigma = \theta_1 \dots \theta_{m+1} = \{x_1/u_1, \dots, x_k/u_k\}$, with $\phi(T_h) = u_h$, $k \geqslant h \geqslant 1$. Then, $T_h = \tau_\lambda(u_h)$, $k \geqslant h \geqslant 1$.

We observe that the clause $C_{m+1} = \phi(C'_{m+1}) = \phi(A') \leftarrow \phi(B'_1) \dots \phi(B'_j)$ is a standardization apart of a clause in $H_\lambda(P)$, with respect to D_1. Then, from G_m in D_1, by exploiting as input clause C_{m+1} and as m.g.u. θ_{m+1}, we obtain a resolvent G_{m+1} such that $G_{m+1} = \phi(G'_{m+1})$. Thus, since $\tau_\lambda(G_{m+1}) = \tau_\lambda(\phi(G'_{m+1})) = G'_{m+1}$ and $\tau_\lambda(C_{m+1}) = \tau_\lambda(\phi(C'_{m+1})) = C'_{m+1}$, the thesis is proved. □

Results in Propositions 5.1 and 5.2 can be clarified by the following simple example.

Example 5.1. Let $P = \{C_1: p(a) \leftarrow r(b);\ C_2: r(a) \leftarrow;\ C_3: h(c) \leftarrow\}$ be a program and $\mathscr{R}$ a similarity such that $\mathscr{R}(a,b) = 0.5$, $\mathscr{R}(a,c) = 0.3$, $\mathscr{R}(c,b) = 0.3$, and 0 in the other cases. Then

$$H_{0.5}(P) = P \cup \{p(a) \leftarrow r(a);\ \ p(b) \leftarrow r(a);\ \ p(b) \leftarrow r(b);\ \ r(b) \leftarrow\}.$$

Moreover, by denoting with $\bar{p} = \tau_{0.5}(p) = \{p\}$, $\bar{r} = \tau_{0.5}(r) = \{r\}$, $\bar{h} = \tau_{0.5}(h) = \{h\}$, $\bar{a} = \tau_{0.5}(a) = \tau_{0.5}(b) = \{a, b\}$, $\bar{c} = \tau_{0.5}(c) = \{c\}$, we have that

$$P_{0.5} = \{\bar{p}(\bar{a}) \leftarrow \bar{r}(\bar{a});\ \ \bar{r}(\bar{a}) \leftarrow;\ \ \bar{h}(\bar{c}) \leftarrow\}.$$

Let us consider, for example, the SLD refutation of $P_{0.5} \cup \{\leftarrow \bar{p}(x)\}$

$$\bar{p}(x) \Rightarrow_{\{x/\bar{a}\}} \bar{r}(\bar{a}) \Rightarrow_{\varepsilon} \square$$

with computed answer substitution $\sigma' = \{x/\bar{a}\}$.
By fixing the well ordering $r \prec p \prec h \prec c \prec b \prec a$, it is easy to verify that there exists a corresponding SLD refutation of $H_{0.5}(P) \cup \{\leftarrow p(x)\}$

$$p(x) \Rightarrow_{\{x/b\}} r(b) \Rightarrow_{\varepsilon} \square$$

with computed answer substitution $\sigma = \{x/b\}$ where $\tau_{0.5}(b) = \bar{a}$ and $\phi(\bar{a}) = b$.

Let us stress that $P \cup \{\leftarrow p(x)\}$ has not SLD refutations, i.e., has not exact solutions.

Finally, we can state some relations which characterize the SLD derivations of the elements in the set $H_\lambda(G_0)$.

Proposition 5.3. *Let $\mathscr{R}$ be a similarity, P a logic program on a first order language $\mathscr{L}$, G_0 a goal. If*

$$D = \tau_\lambda(G_0) \Rightarrow_{C_1,\theta_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m} G_m$$

is a SLD derivation for $P_\lambda \cup \{\tau_\lambda(G_0)\}$ with $\xi = \theta_1 \ldots \theta_m = \{x_1/T_1, \ldots, x_k/T_k\}$, then, for any $G \in H_\lambda(G_0)$, D is also a SLD derivation for $P_\lambda \cup \{\tau_\lambda(G)\}$.

Proof. The thesis directly follows by Proposition 3.2(ii) which states that for any $G \in H_\lambda(G_0)$ it is $\tau_\lambda(G) = \tau_\lambda(G_0)$. □

Proposition 5.4. *Let $\mathscr{R}$ be a similarity, P a logic program on a first order language $\mathscr{L}$, G_0 a goal. If*

$$D = G_0 \Rightarrow_{C_1,\theta_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m} G_m$$

is a SLD derivation for $H_\lambda(P) \cup \{G_0\}$ with $\sigma = \theta_1 \ldots \theta_m = \{x_1/u_1, \ldots, x_k/u_k\}$, then, for any $G \in H_\lambda(G_0)$, there exists a SLD derivation for $H_\lambda(P) \cup \{G\}$

$$D'' = G \Rightarrow_{C_1'',\theta_1''} G_1'' \Rightarrow \cdots \Rightarrow_{C_m'',\theta_m''} G_m'',$$

where $\sigma'' = \theta_1'' \ldots \theta_m'' = \{x_1/t_1, \ldots, x_k/t_k\}$ with $t_h \in H_\lambda(u_h)$, $k \geqslant h \geqslant 1$, $G_i" \in H_\lambda(G_i)$, $C_i" \in H_\lambda(C_i)$, $m \geqslant i \geqslant 0$.

Proof. By Proposition 5.1 there exists a SLD derivation for $P_\lambda \cup \{\tau_\lambda(G_0)\}$

$$D' = \tau_\lambda(G_0) \Rightarrow_{C_1',\theta_1'} G_1' \Rightarrow \cdots \Rightarrow_{C_m',\theta_m'} G_m',$$

where $\sigma' = \theta_1' \ldots \theta_m' = \{x_1/T_1, \ldots, x_k/T_k\}$, with $T_h = \tau_\lambda(u_h)$, $k \geqslant h \geqslant 1$, and $G_i' = \tau_\lambda(G_i)$, $C_i' = \tau_\lambda(C_i)$, $m \geqslant i \geqslant 0$.

By Proposition 5.3, D' is a SLD derivation also for $P_\lambda \cup \{\tau_\lambda(G)\}$, for any $G \in H_\lambda(G_0)$. Then, by Proposition 5.2 there exists a SLD derivation for $H_\lambda(P) \cup \{G\}$

$$D'' = G \Rightarrow_{C_1'',\theta_1''} G_1'' \Rightarrow \cdots \Rightarrow_{C_m'',\theta_m''} G_m'',$$

where $\sigma'' = \theta_1'' \ldots \theta_m'' = \{x_1/t_1, \ldots, x_k/t_k\}$, with $T_h = \tau_\lambda(t_h)$, $k \geqslant h \geqslant 1$, and $G_i' = \tau_\lambda(G_i'')$, $C_i' = \tau_\lambda(C_i'')$, $m \geqslant i \geqslant 0$.

Since it is also $T_h = \tau_\lambda(u_h)$, $G_i' = \tau_\lambda(G_i)$, $C_i' = \tau_\lambda(C_i)$, it follows that $\tau_\lambda(u_h) = \tau_\lambda(t_h)$, $k \geqslant h \geqslant 1$, and $\tau_\lambda(G_i) = \tau_\lambda(G_i'')$, $\tau_\lambda(C_i) = \tau_\lambda(C_i'')$, $m \geqslant i \geqslant 0$. As a consequence, by Proposition 3.2(i) and 3.2(ii) it follows that $t_h \in H_\lambda(u_h)$, $k \geqslant h \geqslant 1$, and $G_i'' \in H_\lambda(G_i)$, $C_i'' \in H_\lambda(C_i)$, $m \geqslant i \geqslant 0$. □

Propositions 5.1–5.4 can be analogously stated for SLD refutations. Thus they imply the computational equivalence between the SLD resolution processes associated to the extended and the abstract programs $H_\lambda(P)$ and P_λ.

6. Overcoming unification failure by similarity

In the previous Section 3 a similarity-based extension of the logic programming paradigm has been introduced. The corresponding procedural semantics can be defined by considering SLD resolution in the extended program $H_\lambda(P)$ and in the abstract program P_λ. In both cases some preprocessing steps are needed. In the next section we will introduce a modified version of the SLD resolution which allows us to compute approximate solutions directly in the given program P, i.e. without any preprocessing step.

This procedure, named similarity-based SLD resolution, exploits a simple variation of the standard unification algorithm that provides the m.g.u. of two atoms. We recall that a mismatch between two function symbols or relation names causes a failure of the unification process. Then, it is rather natural to admit a more flexible unification in which the syntactical identity between function and predicate names is substituted by a similarity $\mathscr{R}$. In other words, if corresponding function/predicate symbols which are different have a non-zero similarity degree in $\mathscr{R}$, the unification process does not fail. A consequence of this assumption is that atoms can be unified with different "tolerance" level of approximation.

As an example, if we consider the atoms $p(a)$ and $q(x)$ and a similarity $\mathscr{R}$ such that $\mathscr{R}(p,q)=0.7$ and $\mathscr{R}(a,b)=0.5$, the substitutions

$$\vartheta_1=\{x/a\}, \quad \vartheta_2=\{x/b\}$$

provides the instances

$$p(a)\vartheta_1 = p(a) \neq q(x)\vartheta_1 = q(a)$$

$$p(a)\vartheta_2 = p(a) \neq q(x)\vartheta_2 = q(b).$$

By assuming that the similarity $\mathscr{R}$ replaces the equality relation, the instances obtained with ϑ_1 and ϑ_2 can be considered "equal", but some "tolerance" must be exploited to overcome the mismatch between the predicate symbols p and q and the constant symbols a and b. In a direct way, a measure of this "tolerance" level can be expressed for ϑ_1 by the similarity value $\mathscr{R}(p,q)=0.7$, and for ϑ_2 by $\mathscr{R}(p,q)\wedge\mathscr{R}(a,b)=0.7\wedge 0.5=0.5$ (i.e., the minimum between the similarity values which relate the mismatching symbols). Intuitively, a higher value of similarity exploited to overcome failures of matching, corresponds to a better value of "tolerance" which is needed to consider "equal" symbols that are different. Thus, it is natural to assume that an acceptable unifier must provide the maximum value of similarity between the obtained instances.

Definition 6.1 formalizes these ideas in the case of first order languages and a generalized unification algorithm is also provided. It is a simplified version of a more complex approach to the notion of similarity-based unification for function free languages which is proposed in [14] by taking into account substitutions of variable with sets of symbols.

Definition 6.1. Given a similarity $\mathscr{R}$, a substitution θ and two atoms $A = p(t_1,\ldots,t_n)$ and $B = q(t'_1,\ldots,t'_n)$ in a first order language, with the same arity. We define the *unification-degree* $v_{\mathscr{R}}(A\theta, B\theta)$ *of* A *and* B, *with respect to* θ *and* $\mathscr{R}$, as follows:

$$v_{\mathscr{R}}(A\theta, B\theta) = \mathscr{R}(A\theta, B\theta) = \mathscr{R}(p,q) \wedge \left(\bigwedge_{i=1}^{n} \mathscr{R}(t_i\theta, t'_i\theta) \right).$$

If the predicate symbols of A and B have different arities, $v_{\mathscr{R}}(A\theta, B\theta) = 0$.

We say that a substitution θ is a *weak unifier of* A *and* B *with unification-degree* λ *up to* $\mathscr{R}$ (in short a λ-unifier) if

$$\lambda = v_{\mathscr{R}}(A\theta, B\theta) = \max_{\varphi \in \Psi} v_{\mathscr{R}}(A\varphi, B\varphi),$$

where Ψ denote the set of all the substitutions.

Two atoms A and B are λ-*unifiable* up to $\mathscr{R}$ if there exists a λ-unifier for A and B with $\lambda \succ 0$, otherwise we say that they are *not unifiable*.

Let us note that if θ is a λ-unifier up to $\mathscr{R}$, since θ does not affect the value $\mathscr{R}(p,q)$, by the previous definition it follows that

$$\lambda = v_{\mathscr{R}}(A\theta, B\theta) = \mathscr{R}(p,q) \wedge \max_{\varphi \in \Psi} \left(\bigwedge_{i=1}^{n} \mathscr{R}(t_i\theta, t'_i\theta) \right).$$

By Definition 2.1(iv) is easy to verify that the following proposition holds.

Proposition 6.1. *Given a strict similarity* $\mathscr{R}$, *a substitution* θ *and two atoms* A *and* B *of a first order language, then* θ *is a classical unifier of* A *and* B *if and only if* $v_{\mathscr{R}}(A\theta, B\theta) = 1$.

In order to extend the notion of most general unifier, a generalized version of the usual pre-order relation between substitutions must be introduced. The following examples of λ-unifiers highlight the problem that can arise.

Example 6.1. Let $\mathscr{R}$ be a similarity such that $\mathscr{R}(p,q) = 0.3$, $\mathscr{R}(a,b) = 0.5$. Given the atoms $p(x)$ and $q(y)$, the substitution $\theta = \{x/y\}$ provides the instances

$$p(x)\theta = p(y), \quad q(y)\theta = q(y)$$

and θ is a weak unifier since it is easy to verify that $v_{\mathscr{R}}(p(x)\theta, q(y)\theta) = \mathscr{R}(p,q) \wedge \mathscr{R}(y,y) = 0.3 \wedge 1 = 0.3$ is the maximum possible unification-degree. On the other hand, the substitution $\sigma = \{x/a, y/b\}$ provides the instances

$$p(x)\sigma = p(a), \quad q(y)\sigma = q(b)$$

and σ is a weak unifier, too, since it is $v_{\mathscr{R}}(p(x)\sigma, q(y)\sigma) = \mathscr{R}(p,q) \wedge \mathscr{R}(a,b) = 0.3 \wedge 0.5 = 0.3$.

In the previous example, since σ provides ground instances, it should be θ more general than σ. However, the classical notion of $\leqslant$ pre-order between substitutions does not hold in this case. Indeed, for any substitution ξ, it is $x\theta\xi = y\theta\xi$, whereas it is $x\sigma \neq y\sigma$.

Thus, in order to extend the classical pre-order between substitutions, we replace the equality between terms in $\mathscr{L}$ with the equivalence relation $\cong_{\mathscr{R},\lambda}$, i.e., the λ-*cut* of $\mathscr{R}$, and we give the following definition.

Definition 6.2. Let θ and σ be two substitutions and $\mathscr{R}$ a similarity. We say that θ is *more general than* σ *at the level* λ *up to* $\mathscr{R}$, denoted with $\theta \leqslant_{\mathscr{R},\lambda} \sigma$, if there exists a substitution ξ such that, for any variable $x \in V$,

$$x\sigma \cong_{\mathscr{R},\lambda} x\theta\xi \quad \text{or, equivalently,} \quad \mathscr{R}(x\sigma, x\theta\xi) \geqslant \lambda.$$

It is easy to verify that the substitutions θ and σ in Example 6.1 satisfy Definition 6.2. Indeed, it results $\theta \leqslant_{\mathscr{R},0.3} \sigma$ since there exists $\xi = \{y/a\}$ such that

$$x\theta\xi = y\xi = a \cong_{\mathscr{R},0.3} a = x\sigma \quad \text{and} \quad y\theta\xi = y\xi = a \cong_{\mathscr{R},0.3} b = y\sigma.$$

Proposition 6.2. *Given* $\lambda \in (0,1]$, *the relation* $\leqslant_{\mathscr{R},\lambda}$ *is a pre-order in the set of substitutions in a first order language* $\mathscr{L}$.

Proof. The reflexivity holds since, for any substitution θ, it is $\theta \leqslant_{\mathscr{R},\lambda} \theta$. Indeed, for any variable $x \in V$, by considering as ξ the empty substitution ε it is $x\theta = x\theta\varepsilon$. Then $\mathscr{R}(x\theta, x\theta\varepsilon) = 1 \geqslant \lambda$, i.e., $x\theta \cong_{\mathscr{R},\lambda} x\theta\varepsilon$.

In order to prove the transitivity, let θ, σ, γ be substitutions such that $\theta \leqslant_{\mathscr{R},\lambda} \sigma$ and $\sigma \leqslant_{\mathscr{R},\lambda} \gamma$. Then there exists ξ_1 and ξ_2 such that, for any $x \in V$,

$$\mathscr{R}(x\sigma, x\theta\xi_1) \geqslant \lambda, \tag{1}$$

$$\mathscr{R}(x\gamma, x\sigma\xi_2) \geqslant \lambda. \tag{2}$$

By Proposition 3.1(i) applied to (1) it follows that, for any $x \in V$,

$$\mathscr{R}(x\sigma\xi_2, x\theta\xi_1\xi_2) \geqslant \lambda. \tag{3}$$

As a consequence, by the transitivity of the similarity in Definition 2.1 applied to (2) and (3), it results that $\mathscr{R}(x\gamma, x\theta\xi_1\xi_2) \geqslant \lambda$. Thus, by considering $\xi = \xi_1\xi_2$, it is $x\gamma \cong_{\mathscr{R},\lambda} x\theta\xi$, and then $\theta \leqslant_{\mathscr{R},\lambda} \gamma$. □

Now we can extend the notion of most general unifier to similarity-based unification by introducing the following definition.

Definition 6.3. Given a substitution θ and two atoms A and B in a first order language with a similarity $\mathscr{R}$. We say that θ is a *weak most general unifier of* A *and* B *with unification-degree* λ *up to* $\mathscr{R}$ (in short a λ-m.g.u.) if the following conditions hold:

(i) θ is a λ-*unifier* of A and B
(ii) $\theta \leqslant_{\mathscr{R},\lambda} \sigma$, for any σ which is a λ-*unifier* of A and B.

Let us stress that, according to the previous definition, all the weak m.g.u. have the same unification-degree, but they can be not equal up to renaming as shown by the following example.

Example 6.2. Let us consider the atoms $A = p(x, y)$ and $B = q(a, z)$, with a similarity $\mathscr{R}$ such that $\mathscr{R}(p, q) = \mathscr{R}(a, b) = 0.3$. It is easy to verify that $\theta_1 = \{x/a, y/z\}$ and $\theta_2 = \{x/b,\ z/y\}$ are both weak m.g.u. of A and B with approximation degree 0.3.

Roughly speaking, the classical notion of equality (up to renaming) of the m.g.u.'s, is replaced for weak m.g.u.'s by a notion of equality modulo the $\cong_{\mathscr{R},\lambda}$ relation (up to renaming), as shown by the following propositions.

Proposition 6.3. *Let A and B be two atoms in a first order language with a similarity $\mathscr{R}$. If $\theta = \{x_1/u_1, \ldots, x_k/u_k\}$ is a weak m.g.u. of A and B with approximation degree λ up to $\mathscr{R}$, then*
(i) *the substitution $\theta' = \{x_1/t_1, \ldots, x_k/t_k\}$ with $t_i \in H_\lambda(u_i)$, $1 \leqslant i \leqslant k$, is a weak m.g.u. of A and B, too.*
(ii) *for any η renaming, $\theta' = \theta\eta$ is a weak m.g.u. of A and B, too.*

Proof. (i) By the hypothesis, it is $\mathscr{R}(A\theta, B\theta) = \lambda$, with λ maximum unification degree. Then, by Proposition 3.1(ii) it follows that $A\theta$ and $B\theta$ are strings of the same length and in corresponding positions they have or equal variable/bracket symbols, or pairs (s, s') of function/predicate symbols such that $\mathscr{R}(s, s') \geqslant \lambda$. Since by the hypothesis it is also $\mathscr{R}(u_i, t_i) \geqslant \lambda$, $1 \leqslant i \leqslant k$, it follows that $\mathscr{R}(A\theta', B\theta') = \lambda$, i.e. θ' is a λ-unifier of A and B.

Moreover, for any λ-unifier σ of A and B, it is $\theta \leqslant_{\mathscr{R},\lambda} \sigma$, i.e. there exists ξ such that $x\theta\xi \cong_{\mathscr{R},\lambda} x\sigma$, for any variable x. By $\mathscr{R}(u_i, t_i) \geqslant \lambda$, $1 \leqslant i \leqslant k$, it follows that, for any x, it is $x\theta \cong_{\mathscr{R},\lambda} x\theta'$. Then, by Proposition 3.1(i) it is also $x\theta\xi \cong_{\mathscr{R},\lambda} x\theta'\xi$. As a consequence, by transitivity it follows that $x\theta'\xi \cong_{\mathscr{R},\lambda} x\sigma$, for any x. Thus, $\theta' \leqslant_{\mathscr{R},\lambda} \sigma$, too.

(ii) By the hypothesis $\mathscr{R}(A\theta, B\theta) = \lambda$, with λ maximum unification degree. Then, by Proposition 3.1(ii), it follows that $\mathscr{R}(A\theta\eta, B\theta\eta) = \lambda$, i.e. $\theta\eta$ is a λ-unifier of A and B.

Moreover, for any λ-unifier σ of A and B, it is $\theta \leqslant_{\mathscr{R},\lambda} \sigma$, i.e. there exists ξ such that $x\theta\xi \cong_{\mathscr{R},\lambda} x\sigma$, for any x. Since η is a renaming, by $\theta' = \theta\eta$ it follows that $\theta = \theta'\eta^{-1}$. Then $x\theta'\eta^{-1}\xi \cong_{\mathscr{R},\lambda} x\sigma$, for any x, i.e. $\theta' \leqslant_{\mathscr{R},\lambda} \sigma$, too. □

Proposition 6.4. *Let A and B be two atoms in a first order language with a similarity $\mathscr{R}$. If θ and θ' are weak m.g.u.'s of A and B with approximation degree λ up to $\mathscr{R}$, then there exists ξ renaming such that $\theta = \{x_1/u_1, \ldots, x_k/u_k\}$ and $\theta' = \{x_1/t_1, \ldots, x_k/t_k\}$ with $u_i\xi \in H_\lambda(t_i)$, $1 \leqslant i \leqslant k$.*

Proof. By the hypothesis there exists ξ and ξ' such that $x\theta\xi \cong_{\mathscr{R},\lambda} x\theta'$ and $x\theta'\xi' \cong_{\mathscr{R},\lambda} x\theta$, for any variable x. Then, by Proposition 3.1(i), it is also $x\theta\xi\xi' \cong_{\mathscr{R},\lambda} x\theta'\xi'$. As a

consequence, by transitivity it implies that $x\theta\xi\xi' \cong_{\mathscr{R},\lambda} x\theta$, for any x. Then ξ and ξ' are renamings. Since for any x it is $x\theta\xi \cong_{\mathscr{R},\lambda} x\theta'$, i.e. $\mathscr{R}(x\theta\xi, x\theta') \geqslant \lambda$, it follows that $\theta = \{x_1/u_1, \ldots, x_k/u_k\}$, and $\theta' = \{x_1/t_1, \ldots, x_k/t_k\}$ with $u_i\xi \in H_\lambda(t_i)$, $1 \leqslant i \leqslant k$. □

A simple modification of the classical unification algorithm given in [2] provides a λ-m.g.u. for two atoms, if they are λ-unifiable, and a negative answer otherwise. With this aim, we shall extend some definitions to the set of equations between terms that can be associated to the atoms to be unified.

Theorem 6.1 (Weak-unification theorem). *Let $\mathscr{R}$ be a similarity in a first order language $\mathscr{L}$. There exists an algorithm* (*called weak-unification algorithm*) *which for any two atoms in $\mathscr{L}$ produces their weak most general unifier up to $\mathscr{R}$ if they are unifiable and otherwise reports nonexistence of a weak unifier.*

Proof. By Definition 6.1, two atoms $A = p(s_1, \ldots, s_n)$ and $B = q(t_1, \ldots, t_m)$ have unification degree not zero only if they have relation symbols with similarity degree $\mathscr{R}(p, q) \succ 0$. It implies that they must have the same arity $n = m$. Then, we can associate to A and B the set of equations

$$W = \{p = q, s_1 = t_1, \ldots, s_n = t_n\}.$$

In order to unify A and B, these equations can be considered as a set of constraints that must be satisfied by substituting variables occurring in the equations with terms. Then, given a substitution θ, we denote the instance of W by θ with $W\theta = \{p = q,\ s_1\theta = t_1\theta, \ldots, s_n\theta = t_n\theta\}$. Since we relaxed the equality constraint with similarity, we can "tolerate" mismatches in the instantiated equations. We call *unification-degree* $\mu_{\mathscr{R}}(W\theta)$ *of W, with respect to θ and the similarity $\mathscr{R}$*, the value

$$\mu_{\mathscr{R}}(W\theta) = \mathscr{R}(p, q) \wedge \left(\bigwedge_{i=1}^{n} \mathscr{R}(s_i\theta, t_i\theta) \right).$$

If $\mu_{\mathscr{R}}(W\theta) \neq 0$, it follows that $\mathscr{R}(s_i\theta, t_i\theta) \succ 0$, $1 \leqslant i \leqslant n$. Then, by Proposition 3.1(ii) the terms $s_i\theta$ and $t_i\theta$, considered as strings of symbols, have in corresponding positions or equal variable/bracket symbols or function/predicate symbols with non zero similarity value.

Denoted with Ψ the set of all the substitutions, we say that a substitution θ is a *weak unifier of W with degree λ up to $\mathscr{R}$* (in short a λ-unifier up to $\mathscr{R}$) if

$$\lambda = \mu_{\mathscr{R}}(W\theta) = \max_{\varphi \in \Psi} \mu_{\mathscr{R}}(W\varphi).$$

Let us note that, since θ does not affect the value $\mathscr{R}(p, q)$, by the previous definition it follows that

$$\lambda = \mu_{\mathscr{R}}(W\theta) = \mathscr{R}(p, q) \wedge \max_{\varphi \in \Psi} \left(\bigwedge_{i=1}^{n} \mathscr{R}(s_i\theta, t_i\theta) \right).$$

Moreover, we say that θ is a *weak most general unifier of W with unification degree λ up to $\mathscr{R}$* (in short a λ-m.g.u.) if the following conditions hold:

(i) θ is a *λ-unifier* of W

(ii) $\theta \leqslant_{\mathscr{R},\lambda} \sigma$, for any σ which is a *λ-unifier* of W.

It is easy to see that the set of equations $W = \{p = q,\ s_1 = t_1, \ldots, s_n = t_n\}$ has the same weak unifiers as the atoms $p(s_1, \ldots, s_n)$ and $q(t_1, \ldots, t_n)$.

We say that a set of equations W_2 *is an improvement* of a set of equations W_1 if the weak unifiers of W_2 are the same of W_1 and for any of these weak unifiers θ it results $\mu_{\mathscr{R}}(W_1\theta) \leqslant \mu_{\mathscr{R}}(W_2\theta)$. When it is both W_1 improved by W_2 and W_2 improved by W_1 we say that these sets of equations are *equivalent*.

A (possibly empty) set of equations is called *solved* if it is of the form $\{x_1 = u_1, \ldots, x_n = u_n\}$ where x_i's are distinct variables and none of them occurs in a term u_j. A solved set of equations determines the substitution $\{x_1/u_1, \ldots, x_n/u_n\}$. This substitution is an unifier of this set of equations with unification degree equal to 1, and clearly it is its m.g.u., that is, it is more general than any other unifier of this set of equations.

Thus, to find a weak m.g.u. of A and B, starting from the set of equations W associated to A and B, we can construct a sequence of improved sets of equations until a solved set is reached. The following algorithm does it if it is possible and otherwise it halts with failure. It provides also the unification-degree U of the obtained weak m.g.u.

WEAK-UNIFICATION ALGORITHM

Given two atoms $A = p(s_1, \ldots, s_n)$ and $B = q(t_1, \ldots, t_n)$ of the same arity with no common variables to be unified, construct the associated set of equation W. If $\mathscr{R}(p, q) = 0$, halts with failure, otherwise, set $U = \mathscr{R}(p, q)$ and $W = W - \{p = q\}$. It is easy to see that the new set of equations is an improvement of the previous one.

Until the current set of equation W does not change, nondeterministically choose from W an equation of a form below and perform the associated action.

(1) $f(s_1, \ldots, s_n) = g(t_1, \ldots, t_n)$ *where* $\mathscr{R}(f, g) \succ 0$: replace by the equations $s_1 = t_1, \ldots, s_n = t_n$, *and set* $U = U \wedge \mathscr{R}(f, g)$;

(2) $f(s_1, \ldots, s_n) = g(t_1, \ldots, t_m)$ *where* $\mathscr{R}(f, g) = 0$: *halts with failure*;

(3) $x = x$: *delete the equation*;

(4) $t = x$ *where t is not a variable*; *replace by the equation* $x = t$;

(5) $x = t$ *where* $x \neq t$ *and x has another occurrence in the set of equations*: *if x appears in t then halt with failure*, *otherwise perform the substitution* $\{x/t\}$ *in every other equations*.

We stress that steps (1), (3)–(5) provide a new set of equations W' which is an improvement of the current set W. In particular, by the reflexivity and symmetry of $\mathscr{R}$ in Definition 2.1, at step (3), (4)–(5) equivalent sets are obtained. At step (1) the constraint expressed by the equality of the two function symbols f and g is removed. Since f and g are not affected by any substitution, the two sets W and W' have the same weak unifiers. The equality constraint between f and g appears in W but not in

W'. Then, by the definition, a weak unifier of W can have a greater unification degree as weak unifier of W'.

The correctness of the previous algorithm can be proven following the same line of the proof given in [2]. □

The following example highlights the modifications introduced in the classical unification algorithm. Let us recall that constant symbols can be considered as functions with zero arity.

Example 6.3. Let us consider a similarity $\mathscr{R}$ such that $\mathscr{R}(p,r)=0.5$, $\mathscr{R}(f,g)=0.7$, $\mathscr{R}(a,c)=0.3$ and the two atoms $p(f(x),a,y)$, $r(g(b),c,f(x))$ to be unified. The unification algorithm given in Theorem 6.1 performs the following steps:

- Since $\mathscr{R}(p,r)=0.5\neq 0$ we consider the set $W_1=\{f(x)=g(b), a=c, y=f(x)\}$ and $U=0.5$.
- Choosing the equation $f(x)=g(b)$ in W_1, since $\mathscr{R}(f,g)=0.7\neq 0$ we have the new set $W_2=\{x=b,\ a=c,\ y=f(x)\}$ and $U=0.5\wedge 0.7=0.5$.
- Choosing the equation $x=b$ in W_2, since x has another occurrence in W_2, we have the new set $W_3=\{x=b,\ a=c,\ y=f(b)\}$.
- Choosing the equation $a=c$ in W_3, since $\mathscr{R}(a,c)=0.3\neq 0$ we have the new set $W_4=\{x=b,\ y=f(b)\}$ and $U=0.5\wedge 0.3=0.3$.
- Since W_4 is a solved set, the computed weak m.g.u. is $\xi=\{x/b, y/f(b)\}$ with $U=0.3$.

Let us verify that ξ has unification degree 0.3

$$
\begin{aligned}
&v_{\mathscr{R}}(p(f(x),a,y)\xi, r(g(b),c,f(x))\xi)\\
&\quad=\mathscr{R}(p,r)\wedge\mathscr{R}(f(b),g(b))\wedge\mathscr{R}(a,c)\wedge\mathscr{R}(f(b),f(b))\\
&\quad=0.5\wedge 0.7\wedge 0.3\wedge 1=0.3.
\end{aligned}
$$

7. Similarity-based SLD resolution

The generalized unification algorithm introduced in the previous section provides an m.g.u. whenever the existence of a nonzero similarity value allows us to overcome failures in the standard unification process. It allows us to modify a Prolog interpreter in order to perform similarity based computations with respect to the extended program $H_\lambda(P)$, or equivalently with respect to the abstract program P_λ, without introducing preprocessing steps.

Indeed, in this section we introduce a modified version of SLD resolution which provides computed answer substitutions with an associated level of approximation expressed by a numeric value $\lambda\in(0,1]$ named *approximation-degree*. This value is given by the minimum unification-degree of the m.g.u.'s exploited in the related derivation. The basic idea of this procedure has been outlined in [16].

In general, a computed answer substitution can be obtained with different SLD refutations and different approximation-degrees, then the maximum of this values characterizes the best refutations of the goal. In particular, a refutation with approximation-degree 1 provides an exact solution. When refutations of a ground atom L are considered, the best value of such approximation degrees provides the value of the membership function $M_{P,\mathscr{R}}(L)$ which characterizes the fuzzy least Herbrand model of the program P.

More formally, we introduce the following generalization of the SLD derivation. Let us stress that this definition can be given also in the case that the Triangular norm $\wedge$ in Definition 2.1 is different from the *minimum*. However, this assumption is needed in order to prove the relations with the extended and abstract programs $H_\lambda(P)$ and P_λ.

Definition 7.1. Given a similarity $\mathscr{R}$, a program P in a first order language and a goal G_0. A *similarity-based* SLD derivation of $P \cup \{G_0\}$, denoted by

$$G_0 \Rightarrow_{C_1,\theta_1,U_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m,U_m} G_m \Rightarrow \cdots$$

consists of a sequence $G_0, G_1, \ldots$ of negative clauses, together with a sequence $C_1, C_2, \ldots$ of variants of clauses from P, a sequence of substitution $\theta_1, \theta_2, \ldots$ and a sequence $U_0, U_1, \ldots$ of values in $[0,1]$, such that for all $i \geqslant 1$:

(i) G_i is a resolvent of G_{i-1} and C_i by using the idempotent m.g.u. θ_i with unification-degree U_i *up to* $\mathscr{R}$,

(ii) C_i has no variables in common with $G_0,\ C_0, \ldots, C_{i-1}$.

When one of the resolvents G_i is the empty clause $\square$, the derivation is called a similarity-based SLD *refutation* up to $\mathscr{R}$. A similarity-based SLD derivation is called *failed* if it is not a refutation.

It is easy to see that when the similarity is the identity, the previous definition provides the classical notion of SLD refutation. We stress that, without loss of generality, the hypothesis of idempotent m.g.u. allow us to consider the static definition of standardization apart given in the statement (ii) of the previous Definition [2].

In the case of a standard SLD refutation, the restriction of $\gamma = \theta_1 \ldots \theta_m$ to the variables of the initial goal G_0 provides a computed answer substitution for $P \cup \{G_0\}$. When a similarity-based SLD refutation is considered, we must take into account that, at any derivation step, the unification algorithm provides the m.g.u. θ_i with an associated unification degree U_i, $i = 1, \ldots, m$. In some sense, these values U_i can be considered as constraints that allowed the success of the unification processes. Then, it is natural to consider the best unification degree that allows us to satisfy all these constraints. More formally, we give the following definitions.

Definition 7.2. Given a similarity $\mathscr{R}$, a program P in a first order language, a goal G_0, and a similarity-based SLD derivation D for $P \cup \{G_0\}$

$$D = G_0 \Rightarrow_{C_1,\theta_1,U_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m,U_m} G_m$$

denoted with $\theta_1 \dots \theta_m = \{x_1/u_1, \dots, x_k/u_k\}$ the restriction of the composition of the m.g.u.'s in D to the variables of G_0, we call *approximation degree associated to* $\theta_1, \dots, \theta_m$ the value

$$\lambda = \bigwedge_{i=1}^{m} U_i.$$

Then, we generalize the definition of computed answer substitution in the frame of similarity-based SLD resolution as follows.

Definition 7.3. Given a similarity $\mathscr{R}$, a program P in a first order language, a goal G_0, and a similarity-based SLD refutation D for $P \cup \{G_0\}$, denoted with $\theta_1 \dots \theta_m = \{x_1/u_1, \dots, x_k/u_k\}$ the restriction of the composition of the m.g.u.'s in D to the variables of G_0 with associated approximation-degree λ, we consider the family $\Psi = \{\sigma_j\}_{j \in I}$ of substitutions such that for any $j \in I$

$$\sigma_j = \{x_1/t_1, \dots, x_k/t_k\} \quad \text{with } t_h \in H_\lambda(u_h),\ k \geqslant h \geqslant 1.$$

Any element in the family Ψ is named *computed answer substitution for* $P \cup \{G_0\}$ *with approximation-degree* λ *up to* $\mathscr{R}$, and is denoted with $\langle \sigma_j, \lambda \rangle$ for any $\sigma_j \in \Psi$.

Let us stress that, when the triangular norm in the Definition 2.1 is the *minimum*, then λ belongs to the set $\{\lambda_1, \dots, \lambda_n\}$ of the possible similarity values in $\mathscr{R}$.

In analogy with the classical case, the similarity-based SLD derivations for $P \cup \{G_0\}$ via a selection rule S can be grouped in a tree structure according to the following definition.

Definition 7.4. Given a similarity $\mathscr{R}$, a logic program P in a first order language, a goal G_0, and a selection rule S. The *similarity-based SLD tree up to* $\mathscr{R}$ *for* $P \cup \{G_0\}$ *via* S is a tree such that

- its branches are similarity-based SLD derivations *for* $P \cup \{G_0\}$ *via* S,
- every node G has exactly one descendant for every clause C of P such that the selected atom A of G unifies with the head of a variant C' of a clause C with unification degree not zero. This descendant is a resolvent of G and C' with A being the selected atom via S in G.

We call a SLD tree *successful* if it contains the empty clause.

Example 7.1. *Let* P be a logic program with the following clauses:

$C_1 := p(a) \leftarrow q.$
$C_2 := q \leftarrow r(f(d)).$
$C_3 := p(b) \leftarrow r(c).$
$C_4 := h \leftarrow.$
$C_5 := r(a) \leftarrow.$

Let us consider a similarity $\mathscr{R}$ in the alphabet of P with the following nonzero values $\mathscr{R}(q,h) = 0.8,\ \mathscr{R}(a,c) = 0.5,\ \mathscr{R}(a,b) = 0.8,\ \mathscr{R}(b,c) = 0.5.$

The SLD tree up to $\mathscr{R}$ of $P \cup \{\leftarrow p(x)\}$ via leftmost is given by

$$
\begin{array}{ccc}
 & p(x) & \\
\Downarrow C_1,\ \{x/a\},\ 1 & \Downarrow C_1,\ \{x/a\},\ 1 & \Downarrow C_3,\ \{x/b\},\ 1 \\
q & q & r(c) \\
\Downarrow C_2,\ \varepsilon,\ 1 & \Downarrow C_4,\ \varepsilon,\ 0.8 & \Downarrow C_5,\ \varepsilon,\ 0.5 \\
r(f(d)) & \square & \square \\
\text{failure} & &
\end{array}
$$

Then the refurations provide the two substitutions

$$\theta_1 = \{x/a\} \quad \text{with approximation degree } \lambda_1 = 1 \wedge 0.8 = 0.8,$$

$$\theta_2 = \{x/b\} \quad \text{with approximation degree } \lambda_2 = 1 \wedge 0.5 = 0.5.$$

According to Definition 7.3, since $H_{0.8}(a) = \{a, b\}$, the family of computed answer substitutions associated to θ_1 is given by $\Psi_1 = \{\{x/a\},\ \{x/b\}\}$. Each solution in this family has approximation degree $\lambda_1 = 0.8$. Analogously, since $H_{0.5}(b) = \{a, b, c\}$, the family of computed answer substitutions associated to θ_2 is $\Psi_2 = \{\{x/a\}, \{x/b\}, \{x/c\}\}$. Each solution in this family has approximation degree $\lambda_1 = 0.5$.

Let us note that the solutions $\{x/a\}$ and $\{x/b\}$ can be obtained with different approximation degrees by different refutations. We also note that, if similarity is not considered, no solution can be obtained.

Propositions 7.1 and 7.2 state the links between the computed answer substitutions obtained by the similarity-based SLD resolution and the solutions obtained by the standard SLD resolution both in $H_\lambda(P)$ and in P_λ. As in Section 5, we are concerned with the leftmost selection rule, but all the presented results can be analogously stated for any selection rule that does not depend on the function and predicate names and by the history of the derivation [2].

At first we prove two technical lemmata.

Lemma 7.1. *Given a similarity $\mathscr{R}$, a program P on a first order language and a goal G_0. If there exists a similarity-based SLD derivation with approximation degree λ up to $\mathscr{R}$ for $P \cup \{G_0\}$*

$$D = G_0 \Rightarrow_{C_1,\theta_1,U_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m,U_m} G_m,$$

where $\sigma = \theta_1 \ldots \theta_m = \{x_1/u_1, \ldots, x_k/u_k\}$, then, there exists a SLD derivation for $H_\lambda(P) \cup \{G_0\}$ with the same m.g.u.'s and resolvents

$$D' = G_0 \Rightarrow_{C'_1,\theta_1} G_1 \Rightarrow \cdots \Rightarrow_{C'_m,\theta_m} G_m,$$

where $C'_i \in H_\lambda(C_i)$, $m \geqslant i \geqslant 1$.

Proof. We prove the thesis by induction on the length of D. If the length of D is zero, then the thesis is true. Let us suppose that the thesis is true for length of D equal m.

Let

$$D = G_0 \Rightarrow_{C_1,\theta_1,U_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m,U_m} G_m \Rightarrow_{C_{m+1},\theta_{m+1},U_{m+1}} G_{m+1}$$

be an existing similarity-based SLD derivation for $H_\lambda(P) \cup \{G_0\}$ of length $m+1$ with approximation degree λ up to $\mathscr{R}$. By Definition 7.2, it is $\lambda = \lambda' \wedge U_{m+1}$, with λ' approximation degree of

$$D_1 = G_0 \Rightarrow_{C_1,\theta_1,U_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m,U_m} G_m.$$

Then, $\lambda \preccurlyeq \lambda'$ and $\lambda \preccurlyeq U_{m+1}$.

By the inductive hypothesis there exists an SLD derivation for $H_{\lambda'}(P) \cup \{G_0\}$

$$D'_1 = G_0 \Rightarrow_{C'_1,\theta_1} G_1 \Rightarrow \cdots \Rightarrow_{C'_m,\theta_m} G_m,$$

where $C'_i \in H_{\lambda'}(C_i)$, $m \geqslant i \geqslant 1$. Since $\lambda \preccurlyeq \lambda'$, by Proposition 2.2(i) it follows that $H_\lambda(P) \supseteq H_{\lambda'}(P)$ and $H_\lambda(C_i) \supseteq H_{\lambda'}(C_i)$. Then, D'_1 is and SLD derivation also for $H_\lambda(P) \cup \{G_0\}$, with $C'_i \in H_\lambda(C_i)$, $m \geqslant i \geqslant 1$.

Let us denote with A the head of the input clause $C_{m+1} = A \leftarrow B_1 \ldots B_j$, and with L the leftmost atom of G_m which is the selected atom both in D_1 and in D'_1. Since θ_{m+1} is a weak m.g.u. of A and L with degree U_{m+1}, then $\mathscr{R}(A\theta_{m+1}, L\theta_{m+1}) = U_{m+1} \geqslant \lambda$. It implies, by Proposition 3.1(ii), that $A\theta_{m+1}$ and $L\theta_{m+1}$ are strings of the same length and, on corresponding positions, they have or equal variables/brackets, or a pair (s, s') of function/predicate symbols such that $\mathscr{R}(s, s') \geqslant \lambda$, i.e., $s' \in H_\lambda(s)$. By substituting to these symbols s in A the corresponding symbol s' in L, we obtain an atom A' such that $A' \in H_\lambda(A)$. Then, the unification algorithm on L and A' emulates the behavior on L and A, provided that mismatching between different function/predicate symbols are substituted with equality relations. As a consequence, θ_{m+1} is a classical m.g.u. of L and A' since the related unification degree is 1.

We observe that, with respect to D'_1, the clause $C'_{m+1} = A' \leftarrow B_1 \ldots B_j$ is a standardization apart of a clause in $H_\lambda(P)$. Then, a derivation step in D'_1 with input clause C'_{m+1} and m.g.u. θ_{m+1} provides the same resolvent G_{m+1} obtained in D. Thus the thesis is proved. □

Lemma 7.2. *Given a similarity $\mathscr{R}$, a program P on a first order language and a goal G_0. If there exists a SLD derivation for $H_\lambda(P) \cup \{G_0\}$*

$$D' = G_0 \Rightarrow_{C'_1,\theta'_1} G'_1 \Rightarrow \cdots \Rightarrow_{C'_m,\theta'_m} G'_m,$$

where $\sigma' = \theta'_1 \ldots \theta'_m = \{x_1/u_1, \ldots, x_k/u_k\}$, then there exists a similarity-based SLD derivation for $P \cup \{G_0\}$ with approximation degree $\lambda' \geqslant \lambda$ up to $\mathscr{R}$,

$$D = G_0 \Rightarrow_{C_1,\theta_1,U_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m,U_m} G_m,$$

where $\sigma = \theta_1 \ldots \theta_m = \{x_1/t_1, \ldots, x_k/t_k\}$ with $t_h \in H_\lambda(u_h)$, $k \geqslant h \geqslant 1$, and $C'_i \in H_\lambda(C_i)$, $G'_i \in H_\lambda(G_i)$, $m \geqslant i \geqslant 1$.

Proof. We prove the thesis by induction on the length of D. If the length of D is zero, then the thesis is true. Let us suppose that the thesis is true for length of D equal m. Let

$$D' = G_0 \Rightarrow_{C'_1,\theta'_1} G'_1 \Rightarrow \cdots \Rightarrow_{C'_m,\theta'_m} G'_m \Rightarrow_{C'_{m+1},\theta'_{m+1}} G'_{m+1}$$

be an existing SLD derivation for $H_\lambda(P) \cup \{G_0\}$ of length $m+1$ where $\theta'_1 \ldots \theta'_m = \{x_1/u'_1, \ldots, x_r/u'_r\}$. By the inductive hypothesis there exists a similarity-based SLD derivation for $P \cup \{G_0\}$ with approximation degree $\lambda'' \geqslant \lambda$ up to $\mathscr{R}$

$$D_1 = G_0 \Rightarrow_{C_1,\theta_1,U_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m,U_m} G_m,$$

where $\theta_1 \ldots \theta_m = \{x_1/t'_1, \ldots, x_r/t'_r\}$, with $t'_h \in H_\lambda(u'_h)$, $r \geqslant h \geqslant 1$, and $C'_i \in H_\lambda(C_i)$, $G'_i \in H_\lambda(G_i)$, $m \geqslant i \geqslant 1$.

Let us denote with A' the head of the input clause $C'_{m+1} = A' \leftarrow B'_1 \ldots B'_j$, and with L' the leftmost atom of G'_m in D'. Since by inductive hypothesis $G'_m \in H_\lambda(G_m)$, the leftmost atom L of G_m is such that $\mathscr{R}(L', L) \geqslant \lambda$. Then, by Proposition 3.1(ii) L' and L are strings of the same length and in corresponding positions they have or equal variable/bracket symbols, or a pair (t', t) of function/predicate symbols such that

$$\mathscr{R}(t', t) \geqslant \lambda. \tag{4}$$

Moreover, since C'_{m+1} is a standardization of a clause in $H_\lambda(P)$, there exists $C_{m+1} = A \leftarrow B_1 \ldots B_j$, standardization of a clause in P, such that $A' \in H_\lambda(A)$ and $B'_l \in H_\lambda(B_l)$, $j \geqslant l \geqslant 1$. Then, always by Proposition 3.1(ii), A' and A are strings of the same length and in corresponding positions they have or equal variables/bracket symbols, or a pair (s', s) of function/predicate symbols such that

$$\mathscr{R}(s', s) \geqslant \lambda. \tag{5}$$

Then, by (4) and (5) and the transitivity of the similarity in Definition 2.1, it follows that the selected atom L in G_m and the head A of the clause C_{m+1} are string respectively equals to L' and A', except for pais (t, s) of corresponding function/predicate symbols such that $\mathscr{R}(t, s) \geqslant \lambda$.

As a consequence, the unification algorithm on A and L emulates the behavior on A' and L' by overcoming failures in the matching between pairs (t, s) of function/predicate symbols in L and A since $\mathscr{R}(t, s) \geqslant \lambda$. Thus, if $\theta'_{m+1} = \{x_1/u_1", \ldots, x/u_n"\}$, the unification algorithm provides for A and L a weak m.g.u. $\theta_{m+1} = \{x_1/t_1", \ldots, x_n/t_n"\}$ with unification-degree $U_{m+1} \geqslant U'_{m+1} \geqslant \lambda$, such that $t_i" \in H_\lambda(u_i")$, $n \geqslant i \geqslant 1$.

This construction and the inductive hypothesis imply that, if $\sigma' = \theta'_1 \ldots \theta'_{m+1} = \{x_1/u_1, \ldots, x_k/u_k\}$, then $\sigma = \theta_1 \ldots \theta_{m+1} = \{x_1/t_1, \ldots, x_k/t_k\}$ with $t_h \in H_\lambda(u_h)$, $k \geqslant h \geqslant 1$, and approximation-degree $\lambda' = \lambda" \wedge U_{m+1} \geqslant \lambda \wedge \lambda = \lambda$.

Thus, a derivation step by G_m in D with input clause C_{m+1} and m.g.u. θ_{m+1} provides a resolvent G_{m+1} such that $G'_{m+1} \in H_\lambda(G_{m+1})$, and the thesis is proved. □

Now we can relate the computed answer substitutions obtained by similarity-based SLD resolution with the solutions obtained by standard SLD resolution both in $H_\lambda(P)$ and in P_λ.

Proposition 7.1. *Given a similarity $\mathscr{R}$, a program P on a first order language, and a goal G_0*

(i) *if $\langle\sigma,\lambda\rangle$ is a similarity-based computed answer substitution for $P\cup\{G_0\}$ with approximation degree λ up to $\mathscr{R}$, then σ is a computed answer substitution for $H_\lambda(P)\cup\{G_0\}$, too.*

(ii) *if $\sigma'=\{x_1/u_1,\ldots,x_k/u_k\}$ is a computed answer substitution for $H_\lambda(P)\cup\{G_0\}$, then there exists a similarity-based computed answer substitution $\langle\sigma_j,\lambda'\rangle$ for $P\cup\{G_0\}$ with approximation degree $\lambda'\geqslant\lambda$ up to $\mathscr{R}$, where $\sigma_j=\{x_1/w_1,\ldots,x_k/w_k\}$ and $w_h\in H_\lambda(u_h)$, $k\geqslant h\geqslant 1$.*

Proof. (i) It follows directly by Lemma 7.1.

(ii) Let

$$D'=G_0\Rightarrow_{C'_1,\theta'_1}G'_1\Rightarrow\cdots\Rightarrow_{C'_m,\theta'_m}\square$$

be a SLD refutation for $H_\lambda(P)\cup\{G_0\}$ with $\sigma'=\theta'_1\ldots\theta'_m=\{x_1/u_1,\ldots,x_k/u_k\}$. Then, by Lemma 7.2 there exists a similarity-based SLD refutation for $P\cup\{G_0\}$ with approximation-degree $\lambda'\geqslant\lambda$ up to $\mathscr{R}$

$$D=G_0\Rightarrow_{C_1,\theta_1,U_1}G_1\Rightarrow\cdots\Rightarrow_{C_m,\theta_m,U_m}\square,$$

where $\sigma=\theta_1\ldots\theta_m=\{x_1/t_1,\ldots,x_k/t_k\}$, with $t_h\in H_\lambda(u_h)$, $k\geqslant h\geqslant 1$.
By Definition 7.3 a computed answer substitutions associate to D is given by $\langle\sigma_j,\lambda'\rangle$, with σ_j in the family $\Psi=\{\sigma_j\}_{j\in I}$ of substitutions such that, for any $j\in I$, $\sigma_j=\{x_1/w_1,\ldots,x_k/w_k\}$ with $w_h\in H_{\lambda'}(t_h)$, $k\geqslant h\geqslant 1$. Since $\lambda'\geqslant\lambda$, by Proposition 2.2(i) it is $H_{\lambda'}(t_h)\subseteq H_\lambda(t_h)$. Moreover, by $t_h\in H_\lambda(u_h)$, $k\geqslant h\geqslant 1$, it is $H_\lambda(t_h)=H_\lambda(u_h)$. Then, $w_h\in H_{\lambda'}(t_h)\subseteq H_\lambda(t_h)=H_\lambda(u_h)$, $k\geqslant h\geqslant 1$, and the thesis is proved. □

Corollary 7.1. *Given a similarity $\mathscr{R}$, a program P on a first order language, and a ground goal G_0. If $\lambda\in(0,1]$ is such that*

$$\lambda=\max\{\gamma\mid \textit{there exists a SLD refutation for } H_\gamma(P)\cup\{G_0\}\} \tag{1}$$

then there exists a similarity-based refutation for $P\cup\{G_0\}$ with approximation degree λ up to $\mathscr{R}$.

Proof. By (1) and Proposition 7.1(ii), there exists a similarity-based refutation D with computed answer substitution $\langle\varepsilon,\lambda'\rangle$ for $P\cup\{G_0\}$ with approximation degree $\lambda'\geqslant\lambda$ up to $\mathscr{R}$, where ε is the empty substitution. By absurd, let us suppose that $\lambda'\succ\lambda$. By Proposition 7.1(i), ε is a computed answer substitution for $H_{\lambda'}(P)\cup\{G_0\}$, too. But

this is in contrast with the hypothesis on λ, then it must be $\lambda' = \lambda$ and the thesis is proved. □

Proposition 7.2. *Given a similarity $\mathscr{R}$, a program P on a first order language, and a goal G_0*

(i) *If $\langle \sigma, \lambda \rangle$, with $\sigma = \{x_1/t_1, \ldots, x_k/t_k\}$, is a similarity-based computed answer substitution for $P \cup \{G_0\}$ with approximation degree λ up to $\mathscr{R}$, then there exists a SLD-refutation for $P_\lambda \cup \{\tau_\lambda(G_0)\}$, with computed answer substitution $\xi = \{x_1/T_1, \ldots, x_k/T_k\}$ such that $T_h = \tau_\lambda(t_h),\ k \geqslant h \geqslant 1$.*

(ii) *If $\xi = \{x_1/T_1, \ldots, x_k/T_k\}$ is a computed answer substitution for $P_\lambda \cup \{\tau_\lambda(G_0)\}$, then there exists a computed answer substitution $\langle \sigma_j, \lambda' \rangle$ for $P \cup \{G_0\}$ with approximation degree $\lambda' \geqslant \lambda$ up to $\mathscr{R}$, where $\sigma_j = \{x_1/w_1, \ldots, x_k/w_k\}$ and $T_h = \tau_\lambda(w_h),\ k \geqslant h \geqslant 1$.*

Proof. (i) By the hypothesis and Definition 7.3, there exists a similarity-based SLD refutation for $P \cup \{G_0\}$

$$D = G_0 \Rightarrow_{C_1,\theta_1,U_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m,U_m} G_m \Rightarrow \square$$

with approximation degree λ and $\sigma' = \theta_1 \ldots \theta_m = \{x_1/u_1, \ldots, x_k/u_k\}$ restriction of the composition of the m.g.u.'s to the variables of G_0, such that $\sigma = \{x_1/t_1, \ldots, x_k/t_k\}$ with $t_h \in H_\lambda(u_h),\ k \geqslant h \geqslant 1$. By Proposition 7.1(i), σ' is a computed answer substitution for $H_\lambda(P) \cup \{G_0\}$, too. Then, by Proposition 5.1, there exists a SLD-refutation for $P_\lambda \cup \{\tau_\lambda(G_0)\}$

$$D' = G' \Rightarrow_{C'_1,\theta'_1} G'_1 \Rightarrow \cdots \Rightarrow_{C'_m,\theta'_m} \square$$

with computed answer substitution $\xi = \theta'_1 \ldots \theta'_m = \{x_1/T_1, \ldots, x_k/T_k\}$ such that $T_h = \tau_\lambda(u_h),\ k \geqslant h \geqslant 1$. Since $t_h \in H_\lambda(u_h),\ k \geqslant h \geqslant 1$, by Proposition 3.2(i) it follows that $\tau_\lambda(t_h) = \tau_\lambda(u_h),\ k \geqslant h \geqslant 1$. Thus the thesis is proved.

(ii) By the hypothesis there exists a SLD derivation for $P_\lambda \cup \{\tau_\lambda(G_0)\}$

$$D' = G' \Rightarrow_{C'_1,\theta'_1} G'_1 \Rightarrow \cdots \Rightarrow_{C'_m,\theta'_m} \square$$

with $\xi = \theta'_1 \ldots \theta'_m = \{x_1/T_1, \ldots, x_k/T_k\}$. Then, by Proposition 5.2, there exists a SLD-refutation for $H_\lambda(P) \cup \{G_0\}$

$$D = G_0 \Rightarrow_{C_1,\theta_1} G_1 \Rightarrow \cdots \Rightarrow_{C_m,\theta_m} \square$$

where $\sigma = \theta_1 \ldots \theta_m = \{x_1/u_1, \ldots, x_k/u_k\}$ with $T_h = \tau_\lambda(u_h),\ k \geqslant h \geqslant 1$. As a consequence, by Proposition 7.1(ii) there exists a computed answer substitution $\langle \sigma_j, \lambda' \rangle$ for $P \cup \{G_0\}$ with approximation degree $\lambda' \geqslant \lambda$ up to $\mathscr{R}$, where $\sigma_j = \{x_1/w_1, \ldots, x_k/w_k\}$ and $w_h \in H_\lambda(u_h),\ k \geqslant h \geqslant 1$. Then, by Proposition 3.2(i) it follows that $\tau_\lambda(w_h) = \tau_\lambda(u_h) = T_h$ and the thesis is proved. □

Corollary 7.2. *Given a similarity $\mathscr{R}$, a program P on a first order language, and a ground goal G_0. If $\lambda \in (0,1]$ is such that*

$$\lambda = \max\{\gamma \mid \textit{there exists a SLD refutation for } P_\gamma \cup \{\tau_\gamma(G_0)\}\} \tag{1}$$

then there exists a similarity-based refutation for $P \cup \{G_0\}$ with approximation degree λ up to $\mathscr{R}$.

Proof. By (1) and Proposition 7.2(ii), there exists a similarity-based refutation with computed answer substitution $\langle \varepsilon, \lambda' \rangle$ for $P \cup \{G_0\}$ with approximation degree $\lambda' \geqslant \lambda$, up to $\mathscr{R}$, where ε is the empty substitution. By absurd, let us suppose that $\lambda' \succ \lambda$. Then, by Proposition 7.2(i), there exists a SLD refutation for $P_{\lambda'} \cup \{\tau_{\lambda'}(G_0)\}$ with ε as computed answer substitution. But it is in contrast with the hypothesis on λ, then it must be $\lambda' = \lambda$. Thus, the thesis is proved. □

As a consequence of Corollary 7.1, similarity-based SLD resolution provides a characterization of the fuzzy least Herbrand model defined in Section 4 as stated by the following result.

Proposition 7.3. *Given a similarity $\mathscr{R}$ and a logic program P on a first order language. For any $L \in B_{\mathscr{L}}$, $M_{P,\mathscr{R}}(L) = \lambda \neq 0$ if and only if λ is the maximum value in $(0,1]$ such that there exists a similarity-based SLD refutation for $P \cup \{\leftarrow L\}$ with approximation-degree λ up to $\mathscr{R}$.*

Proof. By Definition 4.1, $M_{P,\mathscr{R}}(L) = \lambda$ if and only if λ is the maximum value in $(0,1]$ such that there exists an SLD refutation for $H_\lambda(P) \cup \{\leftarrow L\}$. Then, by Corollary 7.1, it follows that $M_{P,\mathscr{R}}(L) = \lambda$ if and only if there exists a similarity-based SLD-refutation for $P \cup \{\leftarrow L\}$ with approximation degree λ up to $\mathscr{R}$. □

By the previous result we obtain an algorithm to compute the fuzzy least Herbrand model. Indeed, for any $L \in B_{\mathscr{L}}$, the membership value $M_{P,\mathscr{R}}(L)$ is given by the best approximation-degree of the refutations in the similarity-based SLD tree for $P \cup \{\leftarrow L\}$. Let us stress that, when the Triangular norm in the Definition 2.1 is the *minimum*, then $M_{P,\mathscr{R}}(L)$ belongs to the set $\{\lambda_1, \ldots, \lambda_n\}$ of the possible similarity values in $\mathscr{R}$.

Finally, let us note that, by Proposition 7.1(ii), it is possible that for a nonground goal G_0 there exists an approximate solution σ w.r.t. some extended program $H_\lambda(P)$ and the similarity-based SLD resolution provides only solutions σ_j with approximation degree $\lambda' \succ \lambda$. In other words, the solution σ with lower approximation degree λ could be missed, as highlighted by the following example.

Example 7.2. Let us consider again the program P and the similarity $\mathscr{R}$ in the Example 5.1. The similarity-based SLD tree for $P \cup \{\leftarrow p(x)\}$ has only one derivation given by

$$p(x) \Rightarrow_{C_1,\{x/a\},1} r(b) \Rightarrow_{C_2,\varepsilon,0.5} \square$$

with composition of the m.g.u. given by $\{x/a\}$ and approximation degree $\lambda' = 1 \wedge 0.5 = 0.5$. The associated family of computed answer substitution is

$$\Psi = \{\langle\{x/a\}, 0.5\rangle, \langle\{x/b\}, 0.5\rangle\}.$$

It is easy to verify that the extended program $H_{0.3}(P)$ has 15 clauses and that the SLD tree for $H_{0.3}(P) \cup \{\leftarrow p(x)\}$ has 9 refutations which provide the following computed answer substitutions

$$\sigma_1 = \{x/a\}, \quad \sigma_2 = \{x/b\}, \quad \sigma_3 = \{x/c\}.$$

All these solutions have the same fixed level $\lambda = 0.3$ of approximation.

Thus, the solution $\sigma_3 = \{x/c\}$ with approximation degree $\lambda = 0.3$ is missed in the similarity-based SLD tree for $P \cup \{\leftarrow p(x)\}$. However, according to Corollary 7.1, there exists a similarity-based SLD refutation for $P \cup \{\leftarrow p(c)\}$ given by

$$p(c) \Rightarrow_{C_1, \varepsilon, 0.3} r(b) \Rightarrow_{C_2, \varepsilon, 0.5} \square$$

with approximation degree $\lambda = 0.3 \wedge 0.5 = 0.3$.

A modified version of similarity-based SLD tree could be introduced, in order to provide all the solution with approximation degree greater that an a priori fixed $\lambda \in (0, 1]$. However, it is natural to expect a consequent overloading in the computation process.

8. Conclusion and future works

The approach to the approximate reasoning followed in this paper exploits formal tools belonging to distinct fields. On one hand, the classical declarative paradigm of the logic programming provides the inference system. On the other hand, the weakening of the equality notion is managed by means of the fuzzy similarity relation. Roughly speaking, it allows us to deal with any vague information that can be represented by identifying different entities at different levels of approximation. In [15] this idea is framed in the logic programming paradigm. A simple and clear semantics for the obtained extension is provided by exploiting exact models of a family of standard logic programs. It is worth stressing that the evaluation structure of this inference system is affected only by the similarity values defined between the elements of the language. In particular, the use of weights on the clauses, of possibility measures on the interpretations, and of fuzzy sets as elements of the language is not requested. Then, two transformation for the given program P, named extended and abstract programs can be defined. The equivalence of the fixpoint semantics of these obtained programs can be proved, and the notion of fuzzy least Herbrand model is also introduced.

Information carried by this extension of the logic programming paradigm can be managed by exploiting standard SLD resolution w.r.t. the extended and abstract program. Since the extended program is obtained by adding new clauses to P, and the

abstract program is on a different language, both these procedures require some time consuming preprocessing steps.

In this paper an extension of the SLD resolution is described, which allows us to avoid these preprocessing steps. The basic idea is to overcome syntactical failures of the refutation process by weakening the equality constraint between function and predicate symbols required in the unification process. This procedure exploits a generalized notion of m.g.u. which allows us to overcome unification failures between different function and predicate symbols if they have a nonzero value of similarity. It leads to a generalization of the classical notion of m.g.u.. The procedural characterization of the fuzzy least Herbrand model has been also discussed.

It is worth stressing that different similarity relations can be considered with respect to the same logic program or declarative database, i.e., the representation of the "exact" knowledge is not affected by the approximations taken into account. To the best of our knowledge, this approach is the first attempt to exploit the similarity relation at a syntactic level in the framework of logic programming, in order to obtain an inference system for approximate reasoning. Moreover, such a result is obtained by means of a very limited overload of the standard procedure. A first release of a Prolog interpreter which implements the proposed similarity-based SLD resolution is in preparation.

Several open questions remain to be investigated. As stressed at the end of Section 7, the definition of similarity-based SLD tree could be modified in order to provide all the possible solutions with approximation degree greater that an a priori fixed $\lambda \in (0,1]$ and the implementation of such a facility could also be considered. Moreover, a nonzero similarity value could also be allowed between functions with different arities. In such a way, the results of functions evaluated during the resolution procedure could be considered in the weak unification process.

The proposed approach can be also framed in the field of deductive database [33]. Then, modified versions of the bottom-up query answering techniques could be investigated. Finally, we recall that the definition of similarity-based SLD resolution has been introduced by considering a general triangular norm $\wedge$. However, the results proved in this paper are strictly connected with the use of the *minimum*. Indeed, only in this case a similarity can be identified with a family of classical equivalence relations. On the other hand, in many applications it is useful to consider the operator $\wedge$ as a different triangular norm (for example: the Lukasievicz product or the usual arithmetic product), providing different notions of similarity. Then, the question arises of giving a semantic framework for logic programming with similarity also in these cases.

Acknowledgements

The author would like to thank Giangiacomo Gerla and the anonymous referees for useful comments and suggestions.

References

[1] M. Abadi, Z. Manna, Temporal Logic Programming, Proc. IEEE Symp. on Logic Programming, 1987.
[2] R.K. Apt, Logic programming, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Vol. B, Elsevier, Amsterdam, 1990, pp. 492–574.
[3] G. Ausiello, R. Giaccio, On-line algorithm for satisfiability problem with uncertainty, Theoret. Comput. Sci. 171 (1997) 3–24.
[4] J.F. Baldwin, S.Q. Zhou, A fuzzy relational inference language, Fuzzy Sets and Systems 14 (1984) 155–174.
[5] M. Baudinet, Temporal Logic Programming is complete and expressive, Proc. ACM Conf. on Principles of Programming Languages, 1989.
[6] L. Biacino, G. Gerla, M. Ying, Approximate reasoning based on similarity, Math. Logic Quart. 46 (2000).
[7] B. Bukles, F. Petry, A fuzzy model for relational databases, Fuzzy Sets and Systems 7 (1982) 213–226.
[8] P. Cousot, R. Cousot, Abstract interpretation and application to Logic Programs, J. Logic Program. 13 (1992) 103–179.
[9] D. Dubois, F. Esteva, P. Garcia, L. Godo, A logical approach to interpolation based on Similarity, Int. J. Approx. Reason. 17 (1997) 1–36.
[10] D. Dubois, H. Prade, Resolution principles in possibilistic logic, Int. J. Approx. Reason. 3 (1990) 1–21.
[11] M.H. van Emden, Quantitative deduction and its fixpoint theory, J. Logic Program. 3 (1) (1986) 37–53.
[12] F. Esteva, P. Garcia, L. Godo, R. Rodriguez, A modal account of Similarity-based reasoning, Int. J. Approx. Reason. 16 (3/4) (1997) 312–344.
[13] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, IEEE Press, Piscataway, NJ, 1995.
[14] F. Formato, G. Gerla, M.I. Sessa, Similarity-based unification, Fund. Inform. 40 (2000) 1–22.
[15] G. Gerla, M.I. Sessa, Similarity in Logic Programming, in: G. Chen, M. Ying, K.-Y. Cai (Eds.), Fuzzy Logic and Soft Computing, Kluwer Acc. Pub., Norwell, 1999, pp. 19–31.
[16] G. Gerla, M.I. Sessa, Similarity logic and similarity PROLOG, Proc. EUROFUSE-SIC'99, 1999, pp. 137–14.
[17] M. Ishizuka, N. Kanai, PROLOG-Elf incorporating fuzzy logic, Proc. 9th Int. Joint. Conf. on Artificial Intelligence, 1985, pp. 701–703.
[18] M. Kifer, A. Li, On the Semantic of rule-based expert systems with uncertainty, Inter. Conf. on Databases Theory 1988, Lecture Notes in Computer Science, Vol. 326, Springer, Berlin, 1988, pp. 186–202.
[19] M. Kifer, V.S. Subrahmanian, Theory of generalized annotated Logic Programming and its applications, J. Logic Program. 12 (3&4) (1992) 335–367.
[20] F. Klawonn, J.L. Castro, Similarity in fuzzy reasoning, Math. Soft Comput. 2 (1995) 197–228.
[21] F. Klawonn, R. Kruse, A Lukasiewicz logic based PROLOG, Math. Soft Comput. 1 (1994) 5–29.
[22] R.A. Kowalski, Predicate logic as a programming language, in: Proc. IFIP'74 (1974) 569–574.
[23] L.V.S. Lakshmanan, F. Sadri, Probabilistic deductive databases, Proc. IEEE Symp. on Logic Programming (1994) pp. 254–268.
[24] R.C.T. Lee, Fuzzy logic and the resolution principle, J. ACM 19 (1972) 109–119.
[25] J.W. Lloyd, Foundations of Logic Programming, Springer, Berlin, 1987.
[26] T.P. Martin, J.F. Baldwin, B.W. Pilsworth, The implementation of FPROLOG a fuzzy PROLOG interpreter, Fuzzy Sets and Systems 23 (1987) 119–129.
[27] M. Mukaidono, Z.L. Shen, L. Ding, Fundamentals of fuzzy PROLOG, Int. J. Approx. Reason. 3 (1989) 179–193.
[28] D. Nauck, F. Klawonn, R. Kruse, Foundations of Neuro-Fuzzy Systems, Wiley, Chichester MA, 1997.
[29] R.T. Ng, V.S. Subrahmanian, Probabilistic Logic Programming, Inform. Comput. 101 (2) (1992) 150–201.
[30] N. Nilson, Probabilistic logic, Art. Intell. 28 (1986) 71–87.
[31] E.H. Ruspini, On the semantics of fuzzy logic, Int. J. Approx. Reason. 5 (1991) 45–88.
[32] M.I. Sessa, Translations and similarity-based logic programming, Soft Comput. 5 (2) (2001).
[33] M.I. Sessa, Flexible querying in deductive database, in: G. Gerla, A. Di Nola (Eds.) School on Soft Computing—Selected Lectures, Series of Studies on Fuzziness and Soft Computing, Springer, Berlin, to appear.

[34] G. Shafer, A Mathematical theory of evidence, Princeton University Press, Princeton, 1976.
[35] Y. Shoham, Reasoning About Change, MIT Press, Cambridge MA, 1988.
[36] V.S. Subrahmanian, On the semantics of quantitative logic, Proc. IEEE Symp. on Logic Programming, 1987, pp. 173–182.
[37] M.S. Ying, A logic for approximated reasoning, J. Symbolic Logic 59 (1994) 830–837.
[38] L.A. Zadeh, Similarity relations and fuzzy orderings, Inform. Sci. 3 (1971) 177–200.